



Provozně ekonomická fakulta

Případová studie č. 2

Unix/Linux

Obsah

1. Historie a vývoj operačního systému

- 1.1. Historie operačního systému Unix
- 1.2. Historie operačního systému Linux

2. Distribuce

- 2.1. Binární distribuce
- 2.2. Zdrojové distribuce
- 2.3. Live distribuce
- 2.4. Mini distribuce
- 2.5. Výběr distribuce

3. Instalace systému

- 3.1. Instalace distribuce Debian
- 3.2. Instalace distribuce Red Hat
- 3.3. Instalace distribuce Slackware

4. Start a konfigurace systému

- 4.1. Start systému
- 4.2. Zavaděč systému
 - 4.2.1. Zavaděč systému LILO
 - 4.2.2. Zavaděč systému GRUB
- 4.3. Adresáře Linuxu

5. Uživatelské rozhraní

6. Systém ovládání souborů

- 6.1. Soubory
- 6.2. Adresáře
- 6.3. Speciální soubory
- 6.4. Práva k souborům
- 6.5. Tabulka datových bloků

7. Systém správy paměti

- 7.1. Jádro
- 7.2. Alokace paměti v jádře
- 7.3. Buddy alokátor na Linuxu
- 7.4. Barvení stránek na FreeBSD

8. Správa procesů

- 8.1. Scheduler

- 8.2. Scheduler na Linuxu 2.4 a nižších
- 8.3. Scheduler na Linuxu 2.5 a 2.6
- 8.4. Scheduler na FreeBSD 4 a nižších
- 8.5. Kernel schedulable entities na FreeBSD 5

9. Adresářový strom

- 9.1. Stromová struktura
- 9.2. Kořenový svazek -root
- 9.3. Adresář /etc
- 9.4. Adresář /dev
- 9.5. Adresář /usr
- 9.6. Adresář /var
- 9.7. Adresář /proc

10. Další vývoj systému

- 10.1. Budoucnost FreeBSD
 - 10.1.1. Management projektu
 - 10.1.2. Technické cíle
 - 10.1.3. Další technické cíle

11. Zdroje použitých materiálů

- 11.1. Texty
 - 11.1.1. Historie a vývoj operačního systému
 - 11.1.2. Distribuce
 - 11.1.3. Instalace systému
 - 11.1.4. Start a konfigurace systému
 - 11.1.5. Uživatelské rozhraní
 - 11.1.6. Systém ovládání souborů
 - 11.1.7. Systém správy paměti
 - 11.1.8. Správa procesů
 - 11.1.9. Adresářový strom
 - 11.1.10. Další vývoj systému

- 11.2. Obrázky

12. Příloha – prezentace

1. Celkový přehled

1.1. Historie operačního systému Unix

Společnosti Bell Telephone Laboratories (divize AT&T) a General Electric začaly roku 1965 pracovat na projektu „MAC of MIT“. Cílem projektu bylo vytvořit operační systém Multics. Později se společnost Bell Telephone Laboratories rozhodla ustoupit od spolupráce. V důsledku toho však neměla k dispozici kvalitní operační systém.

Bell Telephone Laboratories potřeboval operační systém, který by společnosti vyhovoval. Ken Thompson a Denis Ritchie se rozhodli takový systém navrhnout. Ken tento návrh roku 1969 realizoval při vytváření vývojového prostředí na počítači PDP-7. Další výzkumný pracovník společnosti Bell Telephone Laboratories, Brian Kernighan, dal novému operačnímu systému název Unix. Původně se totiž tento operační systém jmenoval Unics, což mělo sloužit jako zesměšnění neefektivního Multicsu.

Později zveřejnil Dennis Ritchie programovací jazyk C. V roce 1973 byl Unix kompletně přepsán z původního Assembleru do jazyka C. V roce 1977 byl operační systém Unix převeden z počítače PDP na nový počítač s použitím procesu, jež se nazývá „porting“. Tato akce byla uskutečnitelná právě proto, že byl operační systém Unix napsán v jazyce C.

Verze byly číslovány V1 až V7. Stručná charakteristika některých verzí:

- ü V1 (1971) - byla napsána v Assembleru, používala se pro práci s texty.
- ü V4 (1973) - byla napsána v jazyce C, umožňovala přenos systému i na jiné architektury.
- ü V6 (1975) - poprvé byla použita mimo Bell Telephone Laboratories, tento systém byl bezplatně předán univerzitám.
- ü V7 (1979) - obsahovala základní sadu příkazů, C překladač atd.

Koncem sedmdesátých let byla společnosti AT&T antimonopolním úřadem zakázána činnost v oblasti počítačového průmyslu. Proto se společnost rozhodla za velmi výhodných finančních podmínek převést licenci na operační systém Unix na některé univerzity. Unix se tedy stal populárním především v akademických kruzích, avšak postupem času se začal prosazovat i v komerční sféře.

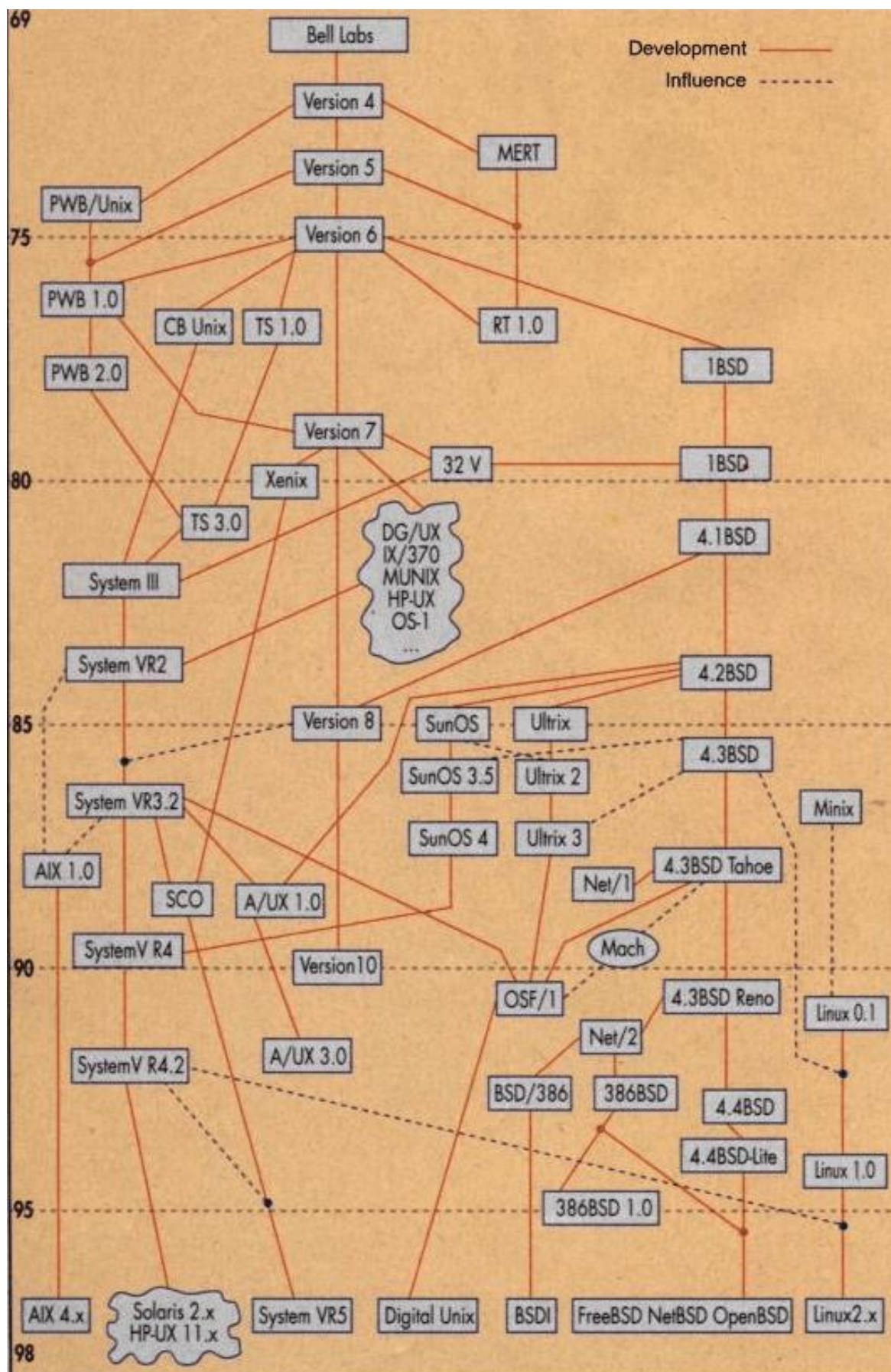
Dnešní podoba Unixu se zcela liší od verze z roku 1970. Existují dvě základní varianty. Jednak je to Systém V od společnosti USL, tedy Unix System Laboratories, který dnes vlastní Novell. A také je to BSD, respektive Berkeley Software Distribution.

Poslední verze USL má označení SVR4, zatímco poslední verze BSD má označení 4.4. Kromě těchto základních verzí však existuje spousta dalších verzí operačního systému Unix. Komerční verze jsou zpravidla odvozeny od jedné z verzí USL nebo BSD. Existuje však spousta verzí operačního systému Unix, které kombinují vlastnosti obou základních verzí.

Základní principy vývoje Unixu:

- ü Psát programy, které budou dělat právě jednu věc a tuto věc budou dělat opravdu dobře.
- ü Psát programy tak, aby mohly navzájem spolupracovat.
- ü Psát programy tak, aby povely přijímaly hromadně ze vstupu v textové podobě
- ü Psát programy tak, aby výstupy produkovaly v textové podobě a mohly být použity jako

vstupy do dalších programů.



Obr. 1 – Vývoj Unixu

1.2. Historie operačního systému Linux

Již roku 1984 začal Richard M. Stallman vyvíjet systém GNU (GNU's Not Unix = GNU není Unix). Jeho cílem bylo vytvoření operačního systému, který nebude zatížen copyrightem. Za tím účelem byla vytvořena speciální licence, která doslova obrací význam copyrightu. Také proto bylo pro vyjádření jejího principu zvoleno pojmenování Dona Hopkinse „copyleft“. Nejstříktnější formou copyleftové licence, která je použita pro všechny zásadní programové části GNU, je GNU GPL (General Public License = Všeobecná veřejná licence). Velice stručně lze GNU GPL shrnout asi takto: předmět licence může být používán, kopírován, pozměňován a distribuován, ale žádná jeho část nesmí být zatížena licencí nekompatibilní s GPL. O něco později byl GNU Project zastřešen FSF (Free Software Foundation = Nadace svobodného software).

Roku 1991 začal student univerzity ve finských Helsinkách **Linus Torvalds** s výzkumem vlastností procesoru Intel 386 a začal pracovat na projektu nového operačního systému pro počítače s tímto procesorem. Začal ho vytvářet jako ročníkový projekt založený na operačním systému Minix profesora Andy Tannebauma, což byl v té době systém pro osobní použití.



Obr. 2 – Linus Torvalds

O svou práci se rozhodl podělit s ostatními a zveřejnil zdrojové kódy svého projektu na internetu pod licencí GNU GPL. Linusova průvodní verze byla v průběhu asi deseti let zdokonalována bezpočtem dalších programátorů z celého světa. A tak vznikl operační systém Linux. Byl pojmenován po Linusovi – jméno **Linux** vzniklo složením jeho jména a písmene X. Všechny verze Unixů totiž podle zvyklosti obsahují písmeno V nebo X.

Když Linus vydával původní verzi Linuxu zdarma na Internet, netušil, že tím založí tak obrovskou vlnu nadšenců pro vývoj software, kterou už asi nic nepřekoná. Dnes pro údržbu nejen jádra Linuxu přispívá tisíce vývojářů, nadšenců, kteří berou Unix (resp. Linux) jako jakýsi „jediný správný systém“.

Zdroje verze 0.01 nebyly spustitelné a o této verzi nebylo nikdy učiněno žádné prohlášení. První oficiální verze systému Linux označovaná jako 0.02 byla ohlášena 5. října 1991. Byla však určená spíš pro nadšence. Obsahovala *bash* a překladač *gcc*. Hlavní pozornost byla zaměřena na jádro, dokumentaci, distribuci... Po verzi 0.03 skočilo číslo verze Linuxu na hodnotu 0.1, protože v systému začalo pracovat více lidí. V březnu roku 1992 byla oficiálně uvedena verze 0.95. V březnu roku 1994 se uvedla verze 1.0.

Protože se Linux neustále vyvíjí, je třeba zavést pořádek do jednotlivých verzí jádra. Označení jádra se skládá ze tří částí. První část označuje číslo verze. Pokud máte verzi 1 a 2, zjistíte, že je v nich podstatný rozdíl. Druhé číslo označuje různé revize jednotlivých verzí. Pokud je číslo sudé, jedná se o stabilní revizi, pokud je číslo liché, jedná se o testovací revizi. Po otestování a opravení chyb je revize prohlášena za stabilní a je jí přiděleno sudé číslo. Třetí číslo pak označuje podverzi revize.

V současnosti Linus pracuje v Santa Clara a pokračuje na údržbě jádra Linuxu, což je základní komponenta operačního systému na nejnižší úrovni. I Windows mají své jádro, ovšem ne v otevřené formě – nejsou tedy k dispozici zdrojové kódy, jak je tomu u Linuxu.

Od jádra 2.0 je s Linuxem spojována postava tučňáka. Původní nápad pochází z dílny Linux Kornel Mailing List. Grafické zpracování tučňáka má na svědomí Larry Swing.

Jméno tučňáka je Tux. Tux je v angličtině zkratka pro „tuxedo“, což je hovorový výraz pro smoking. Černé peří tučňáků přitom vzhledem smoking připomíná. Druhým faktorem je, že Tux má i přímou vazbu na tvůrce Linuxu. Když se rozhodovalo o jménu tučňáka, kdosi řekl, že by to měl být Tux – zkratka pro Torvald's UNIX.



Obr. 3 – Tux

Linus Torvalds: Někteří lidé mi říkali, že tlustý tučňák podle nich moc spanilost Linuxu nevystihuje, a to je pro mne pouze důkazem, že nikdy neviděli rozzlobeného tučňáka, který se na ně řítí rychlostí přes 100 mil za hodinu. Kdyby viděli, dávali by si o hodně větší pozor, co povídají. (*Linus Torvald: Some people have told me they don't think a fat penguin really embodies the grace of Linux, which just tells me they have never seen a angry penguin charging at them in excess of 100mph. They'd be a lot more careful about what they say if they had.*)

Linux je tedy bez omezení šířitelný operační systém unixového typu. Představuje verzi operačního systému Unix pro osobní počítače s procesorem Intel, ale také pro procesory Alpha a další. Byl celý znova vytvořen – na jeho vývoji se společnosti, které vytvořily původní Unixové systémy, tedy Unix System Laboratories a Berkeley Software Distribution, vůbec nepodílely. Zajímavé je, že se na vývoji operačního systému Linux podíleli lidé na celém světě – od Austrálie po Finsko. Všichni doufali, že se Linux podaří uvést do podoby schopné konkurovat ostatním operačním systémům typu Unix.

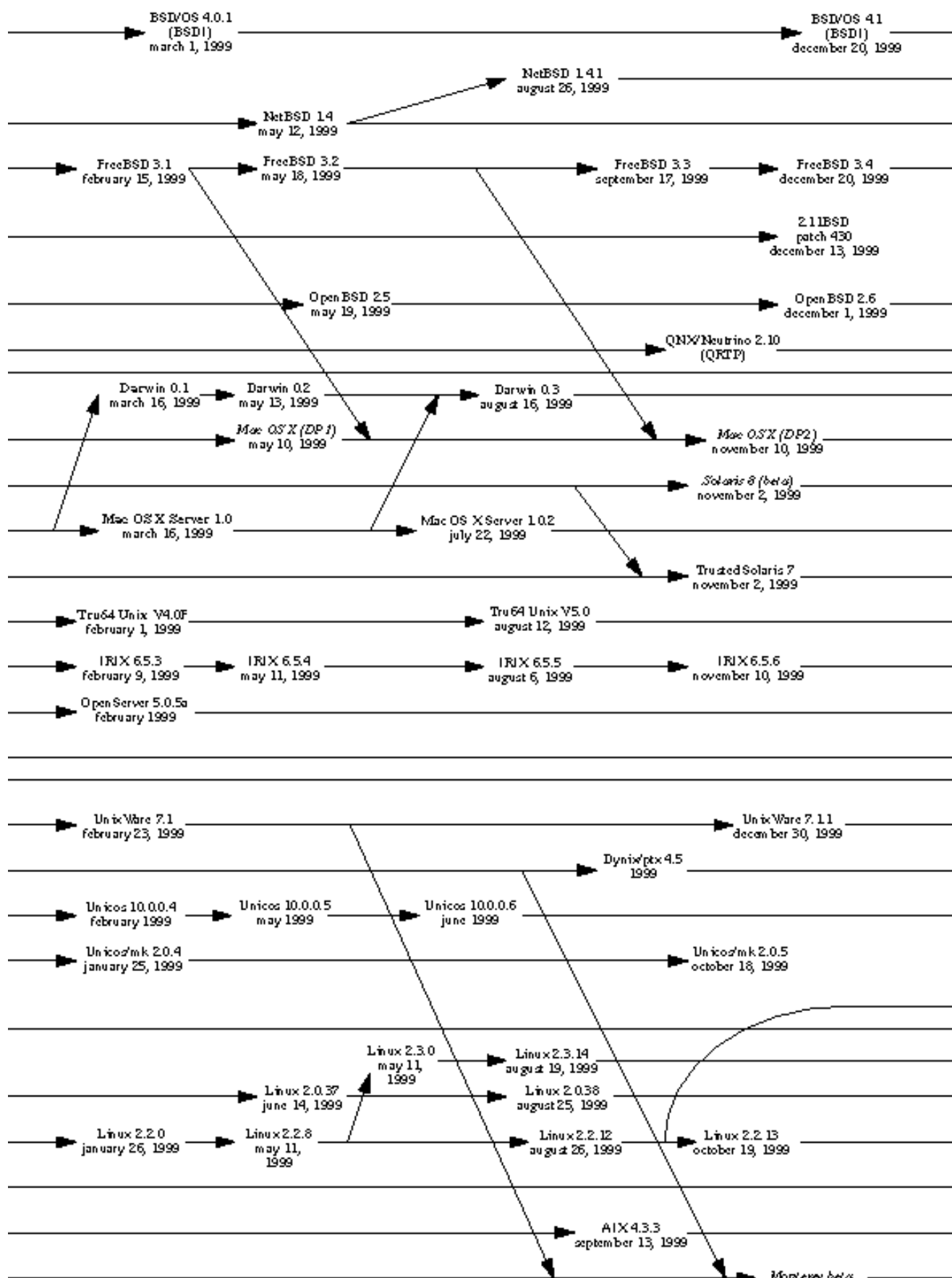
Linux tedy vychází z kořenu Unixu, jehož počátky sahají až do 70. let minulého století. Přestože s původním Unixem nesdílí ani řádku zdrojového kódu, má stejné aplikační rozhraní i filozofii. Díky tomu je kompatibilní s ostatními Unixy na úrovni zdrojových kódů. Proto má i stejné aplikace a nástroje.

Linux podporuje souběžnou práci více uživatelů, z nichž každý může spouštět libovolný počet programů. Linux byl upraven na spoustu jiných platform, než je obvyklé PC. Najdete jej od kapesních počítačů s procesory Arm, přes Macintoshe, pracovní stanice od Sunu až po mainframy od IBM.

Vývoj operačního systému Linux je zcela jedinečný. Při běžné tvorbě software se postupuje tak, že se vytyčí cíle, a předem určená skupina programátorů se těmito cílům snaží co nejvíce přiblížit. Tento systém má tu výhodu, že dopředu lze odhadnout množství potřebného času, náklady a koncovou formu software. Velkou nevýhodou je uzavřená skupina programátorů a nejistota, zda byly cíle určeny správně. Další problém je při hledání chyb, protože menší skupina programátorů často přepisuje celé úseky programu, a tím se tvorba software prodlužuje a prodražuje.

Torvalds se rozhodl, že neponechá vývoj Linuxu jen na svých bedrech, ale podělí se o starosti s širokou programátorskou veřejností. Tak vlastně došlo k velice zajímavému vývoji operačního systému, který probíhá dodnes. Každou část operačního systému má na starosti jeden programátor. Je to jakýsi koordinátor. Toho kontaktuje spousta dobrovolníků, kteří sami vytváří různá vylepšení a doplnění operačního systému. Zjistí-li, že doplnění systému nebo jeho oprava je výhodná, je tato zahrnuta koordinátorem do celku operačního systému.

Tím byl vytvořen jakýsi obrovský programátorský tým, který řeší úkoly daleko rychleji a s mnohem větší silou, než uzavřená skupina programátorů určité softwarové firmy. Je zde však ještě jedna obrovská výhoda. Operační systém Linux není vytvářen s nějakým konkrétním cílem, který nemusí být vždy správný, ale je vytvářen živelně a reaguje na přesné potřeby programátorů a



Obr. 4 – Vývoj OS

správčů sítí. Díky tomu je zaručeno, že operační systém nebude obsahovat zbytečné nesmysly, ale bude obsahovat přesně to, co potřebujete. Navíc všechny části, které se nově vkládají do Linuxu jsou testovány toutéž programátorskou veřejností, která systém vytváří, čímž je zaručena prakticky stoprocentní bezchybnost systému.

Na operační systém Linux se tedy vztahují licenční podmínky GPL (GNU General Public Licence). Tato licence se vztahuje na veškeré produkty, resp. programové vybavení produkované nadací Free Software Foundation. Jejím cílem je zabránit komukoliv omezovat distribuční práva ostatních a zpřístupnit tato práva široké veřejnosti. Software GNU tedy můžeme distribuovat dál, můžeme si za distribuci účtovat kolik chceme, ale nesmíme nikomu nařizovat, za jaký poplatek má distribuci šířit dál. Dále musíme s každou distribucí programového vybavení GNU zpřístupnit také zdrojové kódy. To je velmi užitečné pro programátory, protože si každý může upravit software podle svých vlastních potřeb. Zároveň licence zajišťuje, že se již nikdy nezmění, takže se programátoři nemusí obávat, že by jejich práci mohl někdo zneužít a vytvořit uzavřenou verzi Linuxu šířenou pod jinou licencí.

Linux tak získal velkou pozornost mezi programátory, kteří dlouho očekávali takovýto projekt. Přidali se a začali zasílat opravy a vylepšení. Linux tak rychle zažil prudký vývoj a během několika málo let byl dokončen do podoby, kdy měl všechny základní služby. Od té doby prošel mnoha změnami a dospěl. Již od verze 2.0 může směle konkurovat operačním systémům nižší třídy, např. Windows 95/98, NT, 2000, různým variantám BSD. Od verze 2.4 dokonce většině komerčních Unixů.

Linux je systém, který funguje na mnoha platformách, hlavně je na osobních počítačích řady x86. Podporuje široký rozsah softwaru a hardwaru a to většinou pod licencí GNU. Především podporuje většinu softwaru napsaného pro Unix. Prakticky ale přichází s každou distribucí Linuxu takové množství aplikací, že nemáte potřebu hledat další softwarové vybavení. Pokud přesto vaše distribuce neobsahuje potřebný software, je velká pravděpodobnost, že ho najdete v jiné distribuci Unixu a že bude software ve vašem Linuxu fungovat. Začátkem 90. let prodělal Linux pokles z hlediska zájmu veřejnosti díky nastupujícímu systému Windows NT, ale v dnešní době se vše již obrátilo. Veřejnost i firmy pochopily, že je lepší si zaplatit nebo dokonce nezaplatit za kvalitnější software s velkou podporou.

Velkou roli při vývoji Linuxu sehrál také systém **BSD**. Poskytl důležité nástroje, které učinily Linux populárním. Skupina BSD provedla v návrhu jádra Unixu celou řadu důležitých úprav a tím se verze BSD osamostatnila. Dnes jsou spousty variant systému BSD, například známá FreeBSD. Z jedné varianty systému BSD vychází dokonce operační systém Mac OS X. Navíc i některé síťové utility a démoni v Linuxu jsou také odvozeny z původních zdrojů BSD.



Obr. 5 – Ďáblík BSD

2. Distribuce

2.1. Binární distribuce

Nekomerční distribuce **Debian** GNU/Linux se vyznačuje svým velice konzervativním a pro někoho snad až příliš pomalým vývojovým cyklem a velice zdařilým balíčkovacím systémem *dpkg* s nadstavbou *apt*. Debian bývá často instalován na servery, všude tam, kde je záhodno setkat se s co možná nejmenším výskytem provozních problémů a kde je potřeba garantovat co možná největší míru dostupnosti. Nekomerční povaha je na jedné straně někomu sympatická, ovšem v praxi uzavírá Debianu dveře ve prospěch distribucí s podporou a určitými stupni záruk.



Obr. 6 – Debian

V současnosti velice populární **Mandrake** GNU/Linux vychází z dědictví Red Hatu a je mu také v mnoha ohledech velmi podobný. Jeho tvůrci se orientují především na přizpůsobení distribuce pro využití na desktopech a tedy pro práci v grafickém prostředí. Uživatelé jej obecně považují za pravděpodobně nejkomfortnější linuxovou distribuci, jaká se kdy začala používat.



Obr. 7 – Mandrake

Caldera Network Desktop, **Open** GNU/Linux je komerční distribuce. Obsahuje mimo jiné prostředky pro administraci systému pomocí X11, komerční X Server od firmy X Inside, komerční font server, který rozumí TrueType(tm) fontům, knihovny a vývojové prostředí OSF/Motif a další software. Firma Caldera dodává i další komerční software pro Linux.



Obr. 8 – Caldera

Pokročilejší uživatelé, kteří budou Linux instalovat jak na servery, tak na své domácí počítače, nezdědka sáhnou po nejpopulárnější distribuci vůbec – **Red Hat** GNU/Linuxu. Síla Red Hatu je především v kvalitním zázemí a pozici lídra trhu, kterou si dopřává už několik let. Je to nejrozšířenější distribuce a nabízí zřejmě také nejlepší podporu hardwaru i softwaru od třetích firem. Naopak na druhé straně je potřeba konstatovat, že v určitých oblastech, zejména co se týká konfiguračních nástrojů a asistované instalace balíčků, zůstává v leccems za svými konkurenty.



Obr. 9 – Red Hat

Slackware GNU/Linux se dříve těšil z pozic, které dnes zastává Red Hat Linux. Je jednou z nejstarších distribucí a díky tomu má celkem početnou komunitu uživatelů, kteří jsou na jeho realie zvyklí a nejsou ochotni se jich vzdát. Slackware má image Linuxu, který je nejvíce podobný komerčním Unixům.



Obr. 10 – Slackware

Začátečníkům poskytne nejceněnější služby Linux v podobě krabicové distribuce s instalační příručkou a s dostupnou telefonickou instalační podporou. Takové služby nabízí v našem prostředí nejlépe česká mutace distribuce **SuSE** GNU/Linuxu, která se vyznačuje také výtečnými konfiguračními nástroji, snadnou ovladatelností a zdařilou dokumentací, vše v českém jazyce. SuSE Linux je velice objemná distribuce, najdete v ní téměř všechny programy, které běžný uživatel může kdy potřebovat.



Obr. 11 – SuSE

Tvůrcem komerční distribuce **Turbo GNU/Linux** je společnost Turbolinux se sídlem v americkém San Franciscu. Kromě distribuce Turbo Linux, která je velmi populární v asijských zemích, vyvíjí i vlastní Turbolinux Cluster Server – řešení, které umožňuje kombinovat běžné počítače s operačními systémy Linux, Solaris a Windows NT/2000 do výkonných clusterů. Turbo Linux v České republice distribuuje a doménu turbolinux.cz provozuje firma Danet.cz. Firma poskytuje i technickou podporu.



Obr. 12 – Turbo

2.2. Zdrojové distribuce

Značně populární mezi zdrojovými distribucemi je **Gentoo GNU/Linux**. Sice nemá instalační program, ale instalaci zvládne asi každý, kdo má s GNU/Linuxem alespoň základní administrátorské zkušenosti. Často se doporučuje pro vytížené servery. To ostatně platí o všech zdrojových distribucích, protože je možné je velmi snadno optimalizovat pro konkrétní hardware. Nalezneme zde i vlastní balíčkovací systém Portage.



Obr. 13 – Gentoo

Další známou zdrojovou distribucí je **Source Mage GNU/Linux**. Vychází ze dvou dřívějších projektů – Sorcery GNU/Linux a Lunar GNU/Linux. Výhodou je, že při aktualizaci stahuje balíčky z nativních adres a ne tedy například až odněkud z Kanady nebo Austrálie, když totéž je k dispozici i na mirroru ČVUT.

2.3. Live distribuce

Jako live distribuce označujeme ty, které můžeme spustit například rovnou z CD. Stačí tedy jen vložit CD do mechaniky, restartovat počítač a nastavit čtení z CD-ROM a spustí se nám GNU/Linux. Konfigurační soubory jsou uloženy na disketě, pevném disku nebo v operační paměti. Nemusíme jej tedy instalovat na disk, nijak zvlášť složitě konfigurovat apod.

Tyto linuxové distribuce jsou vhodné tam, kde nemáme plný přístup k systému a přesto z něj chceme dostat maximum. Stejně tak se ale hodí i pro vyzkoušení nového operačního systému a stručné seznámení s ním.

Jednou z live distribucí je **AdiOS GNU/Linux**. Slouží zejména pro školství a vzdělávání, řada programů je využitelná jako užitečná pomůcka při výuce.

Projekt **DaNiX GNU/Linux** je českou modifikací Knoppixu. Disponuje českým prostředím a lokalizovanými programy. Cílem tohoto projektu, který byl začleněn do *edunix.cz* jako řešení pro desktopové uživatele, je především vytvoření snadno ovladatelné a univerzální linuxové live distribuce.



Obr. 14 – Danix

Knoppix GNU/Linux si získal obdiv a přízeň všech, kteří kdy live distribuce využívali. Jedná se bezpochyby o to nejlepší a nejpropracovanější, co na poli tohoto typu distribucí bylo prozatím vytvořeno. Knoppix je populární především díky velice snadnému ovládání, kterým disponuje a díky téměř 2 GB programů a aplikací, jež jsou na CD zkomprimovány a můžeme je plně využívat. Jedná se o skvělou distribuci především pro ty, kteří chtějí získat plnohodnotný GNU/Linux bez toho, aby si jej instalovali na disk.



Obr. 15 – Knoppix

Další live verzí je **Morphix** GNU/Linux. Jedná se v podstatě o Debian GNU/Linux ve formě live distribuce. Na rozdíl od Knoppixu se více drží původního Debianu a není moc vhodný pro začátečníky.



Obr. 16 – Morphix

Live distribuce **Movix** GNU/Linux slouží primárně především k přehrávání multimédií a může být i součástí samotného CD s filmem.



Obr. 17 – Movix

Slax GNU/Linux je bootovatelné „Live CD“ s operačním systémem Linux. Běží přímo z CD media bez nutnosti instalace na pevný disk. Live CD je založené na Slackware Linuxu a je možné jej stáhnout jako ISO. Navíc jsou k dispozici také všechny skripty a zdrojové kódy potřebné pro vytvoření vlastní LiveCD distribuce.



Obr. 18 – Slax

2.4. Mini distribuce

Existuje celá řada distribucí, které se vejdou například na jednu nebo jen několik málo disket. Zpravidla obsahují malý linuxový kernel, základní nástroje pro práci v sítích apod. Nebývají obvykle využívány přímo k uživatelské práci, protože neobsahují dostatečné množství nástrojů a nutný „komfort“ pro práci se systémem. Zpravidla mívají nějaký svůj účel, kterému je sestavení celé distribuce podřízeno.

V dnešní době jsou minidistribuce prakticky vytlačeny díky live distribucím anebo se alespoň kombinují dohromady. Obvykle i mini distribuci nalezneme na CD. Těch klasických „disketových“ v použitelném stavu je dnes už velmi málo.

Jednou z mini distribucí je **Brutalware** GNU/Linux. Tato distribuce je umístěna na třech disketách a je určena pro použití tam, kde nemáte plný přístup k nastavení PC (škola, internetová kavárna apod.) a přesto chcete využívat co nejvíce z jeho výkonu. Kromě toho, že zvládá komunikaci přes protokol PPP, síťové služby a mnoho dalšího, obsahuje také některé zajímavé programy, jako jsou sniffery, scannery atd. Ačkoli se jedná už o poměrně starou distribuci a její vývoj nepokračuje, má stále své příznivce.

Distribuce **Brutalware II** GNU/Linux se zatím pouze vyvíjí. Můžeme však doufat, že z něj vzejde kvalitní následovník původního Brutalwaru. Jedná se vlastně o kombinaci live a mini distribuce. Brutalware II bude umístěn na mini CD a obsah bude tvořit samozřejmě základní uživatelské rozhraní a exploity, sniffery, wifi utility apod.

Damn Small GNU/Linux je pomezí mezi live a mini distribucí. Je kompletně postavena na Knoppixu (respektive na Debian GNU/Linuxu) a nabízí i podobné služby. Rozdíl je ale v tom, že obsahuje pouze základní nástroje pro práci se systémem.

Monkey – Mini Linux je miniaturní distribuce Linuxu na 5 disket. Distribuce se instaluje na DOSovou partition do adresáře *LINUX*. Může sloužit nejen jako malý Linux na hraní.

2.5. Výběr distribuce

Vzhledem k povaze šíření Linuxu je základní existenční jednotkou distribuce neboli společně šířená ucelená sada jádra Linuxu, systémových knihoven, utilit, uživatelských nástrojů a aplikací. Jednotlivé programy se v distribuci nacházejí v podobě balíčků. Balíček je program, který

je připraven k instalaci. Obsahuje vlastní programové soubory, nastavovací skripty a informace o závislosti programu v na jiných programech nebo nastavení systému. Výhodou balíčkových systémů je, že nejste nuceni si nainstalovat co nechcete, ale nainstalujete si jen co potřebuje.

V současné době se nejčastěji setkáváme se dvěma typy balíčků. Prvním jsou RPM, dříve Red Hat Package Manager a později RPM Package Manager, které používají distribuce Red Hat, SuSE či Mandrake. Druhým jsou balíčky DEB, nalézající se v distribucích odvozených od Debianu. Teoreticky by mělo být možné používat RPM balíčky ve všech distribucích založených na RPM, v praxi to tak bohužel nefunguje. Vzhledem k odlišnostem v jednotlivých distribucích, jinému schématu závislostí a dalším nekompatibilitám lze očekávat bezproblémovou funkčnost pouze u balíčků připravených pro odpovídající distribuci. V současné době se objevují standardizační snahy usilující o změnu tohoto nepříjemného stavu.

Jednotlivé **distribuce se od sebe liší** především v:

- Skladbě programů.
- Frekvenci a způsobu vydávání aktualizací.
- Instalačním programem a konfiguračních nástrojích.
- Řešení startovacích skriptů a jejich obsahu.
- Detailch organizace adresářů na disku.
- Dodatečné úpravě některých programů a zejména jádra.
- Ceně a poskytovaných službách přidané hodnoty - dokumentace, technická podpora atd.

Výběr vhodné distribuce je poměrně složitý. Jak již jsme si uvedli výše, **parametry distribucí** jsou poměrně odlišné:

- Debian je podle některých názorů až moc pomalu se vyvíjející, ale přitom k dostání free. Používá balíčky DEB.
- Mandrake je nejvhodnější pro přechod z operačního systému Windows. Objevují se však také názory, že Mandrake je již velmi podobný MS Windows.
- Red Hat je poměrně komerční
- Slackware je pro velmi pokročilé uživatele, kteří už vědí co, kde a jak.
- SuSe je částečně komerční a některé distribuce jsou velmi drahé, ale má dobrou podporu a pravidelný vývoj. Používá balíčky RPM.

Pro uživatele-začátečníka, který se chce s Linuxem něco naučit, je pravděpodobně nejlepší volbou pravděpodobně SuSE nebo Debian.

3. Instalace systému

3.1. Instalace distribuce Debian

Debian lze nainstalovat prakticky z jakéhokoli média. Nejběžnější způsoby jsou dva:

Ü Přes FTP nebo NFS:

Oficiální zdroj distribuce se nachází na: <ftp://ftp.debian.org>

České zdroje distribuce se nachází na: <ftp://sunsite.mff.cuni.cz/MIRRORS/ftp.debian.org> nebo <ftp://ftp.fi.muni.cz/pub/linux/distributions/debian>

Ü Z CD:

Existuje tzv. *Debian/GNU Linux Official CD* vytvořené z autorizovaných image. Toto CD prodává řada firem, můžete si je také sami stáhnout z <ftp.debian.org> nebo odjinud a vypálit.

Oficiální součástí Debianu nejsou non-free programy. Ty se nachází v jiných adresářích na FTP nebo na dodatečných CD dodávaných prodejcem společně s oficiální sadou.

Pokud uživatel nemá bootovací CD nebo potřebuje speciální jádro, musí vytvořit bootovací disketu. Odpovídající image nalezne v adresáři *stable/disks-i386*. Z diskety se nabootoje a dále se postupuje podle instrukcí. Ty však v současné době existují pouze v angličtině a některých dalších evropských jazycích, česká verze instalace zatím neexistuje.

Po nainstalování nejzákladnějších komponent systému z disket nebo jim odpovídajícího *tar* souboru je uživatel vyzván k rebootu. Po novém naboootování, tentokrát již z disku, se dostane do package manageru, kde si zvolí balíky, které chcete instalovat. Následně instalaci dokončí.

3.2. Instalace distribuce Red Hat

Distribuce Red Hat je dostupná z <ftp.fi.muni.cz> a SunSITE.cz. Red Hat se instaluje poměrně jednoduše. K instalaci z CD nebo přes NFS je potřeba jedna bootovací disketa, k instalaci přes FTP nebo s PCMCIA zařízením jsou nutné diskety dvě.

Soubory s obrazem obou disket *boot.img* a *supp.img* je nutné uložit na dvě diskety. Pod DOSem k tomu lze použít například program *rawrite.exe*, pod Unixem stačí příkaz *dd if=soubor.img of=/dev/fd0*.

Poté již stačí vložit disketu *boot.img* do systémové mechaniky, naboootovat z ní a řídit se instrukcemi. Nejprve je nutné odpovědět na několik otázek ohledně hardwaru a způsobu instalace. Poté je do paměti natažena případně i druhá instalační disketa a vlastní instalace může začít.

Při instalaci přes FTP nebo NFS je nejprve nutno nakonfigurovat síť. Jádro musí rozpoznat síťovou kartu – je nutné určit druh karty a případně další parametry. Co se týče parametrů sítě, **musí se nastavit** ještě:

Ü Vlastní IP adresa.

Ü Síťová maska – obvykle 255.255.255.0.

Ü Případná adresa routeru - IP adresa počítače, který spojuje síť uživatele se sítí FTP nebo NFS serveru.

Ü Adresa DNS serveru – pro překlad IP adres na jména.

Ü Jméno FTP nebo NFS serveru.

Dále je nutné zadat adresář na serveru, odkud se bude instalovat. Musí se vložit plná cesta k adresáři, ve kterém je umístěn adresář *RedHat*.

Další částí instalace je rozdělení pevného disku na oblasti: Pro Linux potřebujete minimálně dvě oblasti:

- **Odkládací prostor** – lze případně sdílet s jinými operačními systémy, jeho velikost se volí zhruba 2x až 4x větší než je velikost operační paměti.
- **Souborový systém** – velikost se liší podle toho, co chceme s Linuxem dělat. Instalace samotného systému zabere 40 – 300 MB diskového prostoru, dále je nutné mít prostor pro vlastní data.

Toto se provede programem *fdisk*, který je spuštěn v průběhu instalace. Je pochopitelně nutné mít příslušné volné místo na disku, neobsazené oblastí jiného systému. Pro zmenšení již existující oblasti se souborovým systémem FAT je zde program *fips.exe* – tento program je nutné spustit ještě před vlastní instalací Linuxu.

Závěrem se již pouze vybírá, jaké části systému chceme nainstalovat a poté stačí spustit vlastní instalaci. Po skončení kopírování dat je nutné nakonfigurovat zavaděč systému, program LILO, zadat superuživatelské heslo a poté již je možné nabootovat z disku RedHat Linux.

3.3. Instalace distribuce Slackware

Nejprve je opět nutné zvolit nějaký FTP server, který poskytuje distribuce Slackware, u nás například <ftp://ftp.fi.muni.cz/pub/linux/distributions/slackware>.

V podadresáři *bootdsk.144* zvolíme jeden ze souborů, který obsahuje bootovací disketu. Pokud nemáte SCSI ani síť, volte *bare.gz*. Jinak *scsi.gz*, *scsinet* nebo *net*.

V adresáři *rootdsk.144* vybereme disketu s kořenovým souborovým systémem. Pro barevnou instalaci soubor *color144.gz*, pro instalaci bez menu zvolíme soubor *tty144.gz*. Pomocí programu *dd* dostaneme soubory na diskety: *dd bs=18k if=soubor of=/dev/fd0*. Pod MS-DOSem můžeme použít *Rawrite.exe* z adresáře *install*.

Ostatní instalační diskety vytvoříme zkopírováním jednotlivých adresářů z adresáře *slackware* (nikoliv *slackware*, ale *slakware*!) na klasické, MS-DOSovsky formátované diskety. Tyto adresáře obsahují **sady softwaru**, které je možno instalovat. Jsou to:

- *a* Základní systém
- *ap* Aplikace – *ghostscript*...
- *d* Development – *gcc*...
- *e* Emacs
- *f* FAQ – často kladené dotazy, HOWTOs...
- *i* Info – soubory pro *texinfo*
- *iv* InterViews
- *n* Networking
- *oop* Smalltalk
- *q* Různá předkompilovaná jádra systému
- *t* *TeX*, *Metafont*, *xdvi*, *dvips*...

- *tcl* tcl, tclX, tcl/tk
- *x* X Window System – XFree86 + spousta programů
- *xd* X Window System development – statické knihovny pro X, vývojové prostředky
- *xap* X-aplikace – *ghostview*, *xv*...
- *xv* Xview toolkit a OpenLook window manager – graf. rozhraní pro X
- *y* Hry – např. hra *sasteroids*

Nyní stačí nabootovat z bootdisku vytvořeného např. ze souboru *scsi.gz* a pak se již řídit instrukcemi.

4. Start a konfigurace systému

4.1. Start systému

Po spuštění počítače inicializuje BIOS obrazovku počítače a klávesnici a provede test paměti. Protavím nejsou zjišťována ani kontrolována žádná externí zařízení ani žádný software v počítači.

Teprve když systému dokončí setup, začne pracovat dále. Nejprve si z paměti CMOS zjistí informace o aktuálním datu a čase. Následně dojde k nahrání informací o základních perifériích. Jakmile bude přečtena paměť CMOS, měl by již počítač vědět o tom, jaký disk je v něm přítomen, jak je velký a jakou má strukturu. V tuto chvíli už nic nebrání spuštění operačního systému.

Samotné spuštění systému je ale trochu složitější. Jako první se nahraje 512 bytů dlouhý sektor (MBR – Master Boot Record) z disku do paměti a spustí se program na jeho začátku. Tento program tvoří rozhodující krok v dalším startování systému, neboť určuje, který operační systém se spustí – tento program se nazývá Zavaděč systému.

Kromě *Master Boot Recordu* existují ještě tzv. *Boot Recordy*, které nalezneme na začátku každého diskového oddílu. Právě pomocí nich spouštíme daný operační systém. K těmto *Boot Recordům* přistupujeme právě z MBR.

4.2. Zavaděč systému

Zavaděč systému (bootstrap leader) je program, pomocí kterého se Linux spouští. Po spuštění počítače uvítá uživatele obrazovka, na níž si vybere, co vlastně chce spustit – např. zda Linux, MS Windows apod.

Instalací zavaděče docílíme toho, že se část jeho kódu nahraje do MBR a místo okamžitého spuštění některého operačního systému se nám spustí právě jeho prostředí. V něm následně zvolíme, který systém chceme spustit. Podmínkou však je, že zavaděč, který použijeme, musí umět správně pracovat se všemi dostupnými operačními systémy, které chceme spouštět. Tuto informaci získáme v dokumentaci příslušného zavaděče.

V Linuxu jsou nejrozšířenějšími zavaděči systému LILO a GRUB.

4.2.1. Zavaděč systému LILO

LILO neboli Linux Loader je rozhodně nejznámějším a asi nejpopulárnějším systémovým zavaděčem, který byl kdy v Linuxu použit. Jeho instalace se obvykle provádí do MBR. LILO má přístup k diskovým oddílům a je schopen najít na nich potřebné soubory pro spuštění daného systému.

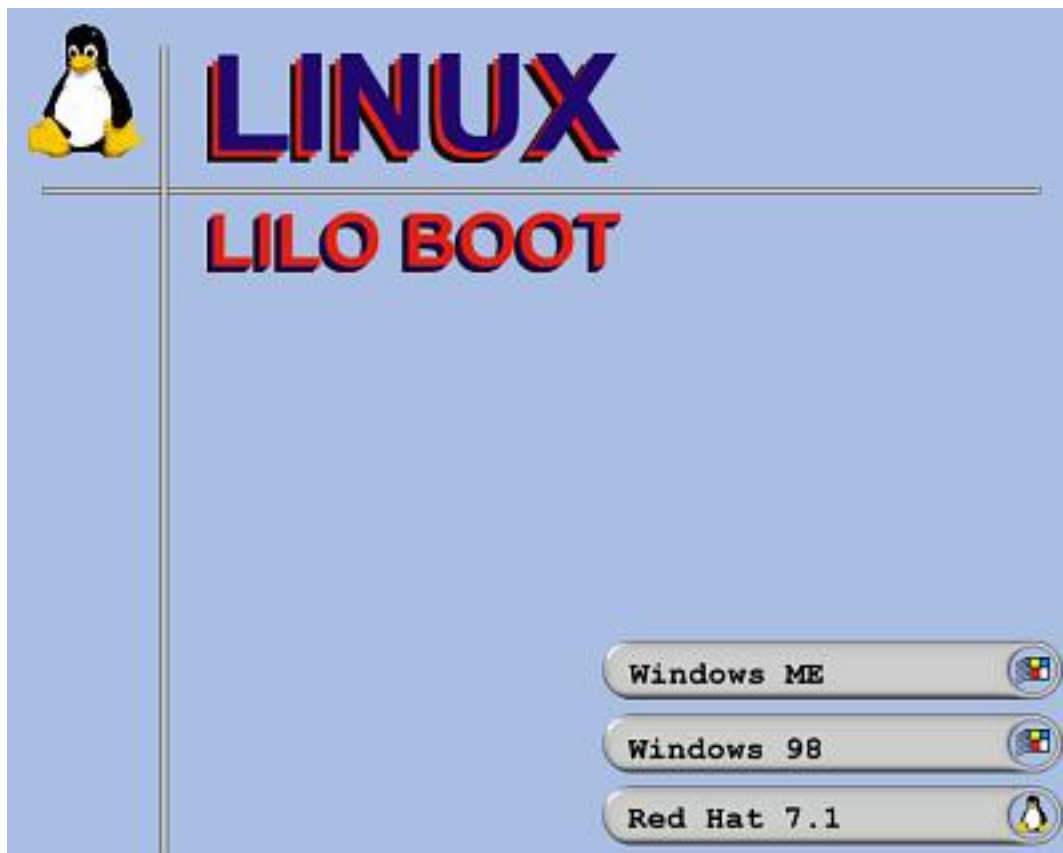
Kromě toho, že můžeme spustit kterýkoli z vypsáných operačních systémů, nabízí nám LILO také možnost předávání instrukcí linuxovému jádru pomocí speciálního příkazového řádku. Z důvodu ochrany systému může být použit LILO také chráněn heslem.

Podíváme se na tento zavaděč z praktického hlediska. Při instalaci většiny distribucí se nakonfiguruje a uloží sám. Nemusíme si s ním tedy lámat hlavu. Například právě SuSE Linux

nabízí v závěru instalace automatické nastavení zavaděče systému. Stačí tedy klepnout myší a je to.

Co ale dělat, když potřebujeme provést změnu. Některé distribuce nabízejí propracované grafické nástroje, pomocí nichž můžeme LILO nakonfigurovat několika klepnutími myší. U těch ostatních je to ale malinko složitější. Podrobný popis konfigurace linuxového zavaděče LILO lze nalézt v doprovodném manuálu.

LILO je velmi populárním a rozšířeným zavaděčem pro Linux. Pokud však uživatel chce používat Linux na počítači s MS Windows XP, není LILO vhodnou volbou, protože s MS Windows XP pracovat neumí.



Obr. 19 – Jedna z možných variant vzhledu startovací nabídky LILO

4.2.2. Zavaděč systému GRUB

V poslední době se tento zavaděč dostává do stále většího podvědomí. GRUB začal suverénně vytlačovat LILO a ohrožuje jeho vůdčí pozici na tomto poli. Ačkoliv z uživatelského hlediska se jedná o zavaděč velice podobný, disponuje řadou „vymožeností“, které LILO doposud postrádá.

Jedná se především o schopnost práce s jednotlivými souborovými systémy a také o možnost úpravy konfiguračních souborů bez následné reinstalace zavaděče.

Dalším podstatným rozdílem je to, že LILO je od základu vytvořeno v šestnáctibitovém Assembleru, zatímco GRUB je naprogramován v přenositelném dvaatřicetibitovém C.



Obr. 20 – Jedna z možných variant vzhledu startovací nabídky GRUB

4.3. Adresáře Linuxu

Po nainstalování Linuxu se na disku vytvoří několik adresářů. Pro ty kteří jsou zvyklí na OS Microsoft Windows to je většinou trochu matoucí. Jejich struktura je však zcela logická.

Všechny disky a diskové oddíly jsou v jedné stromové struktuře. Nejvyšší adresář ve stromě je kořenový adresář, který se zapisuje pomocí lomítka (/).

Základní přehled adresářové struktury v systému LINUX:

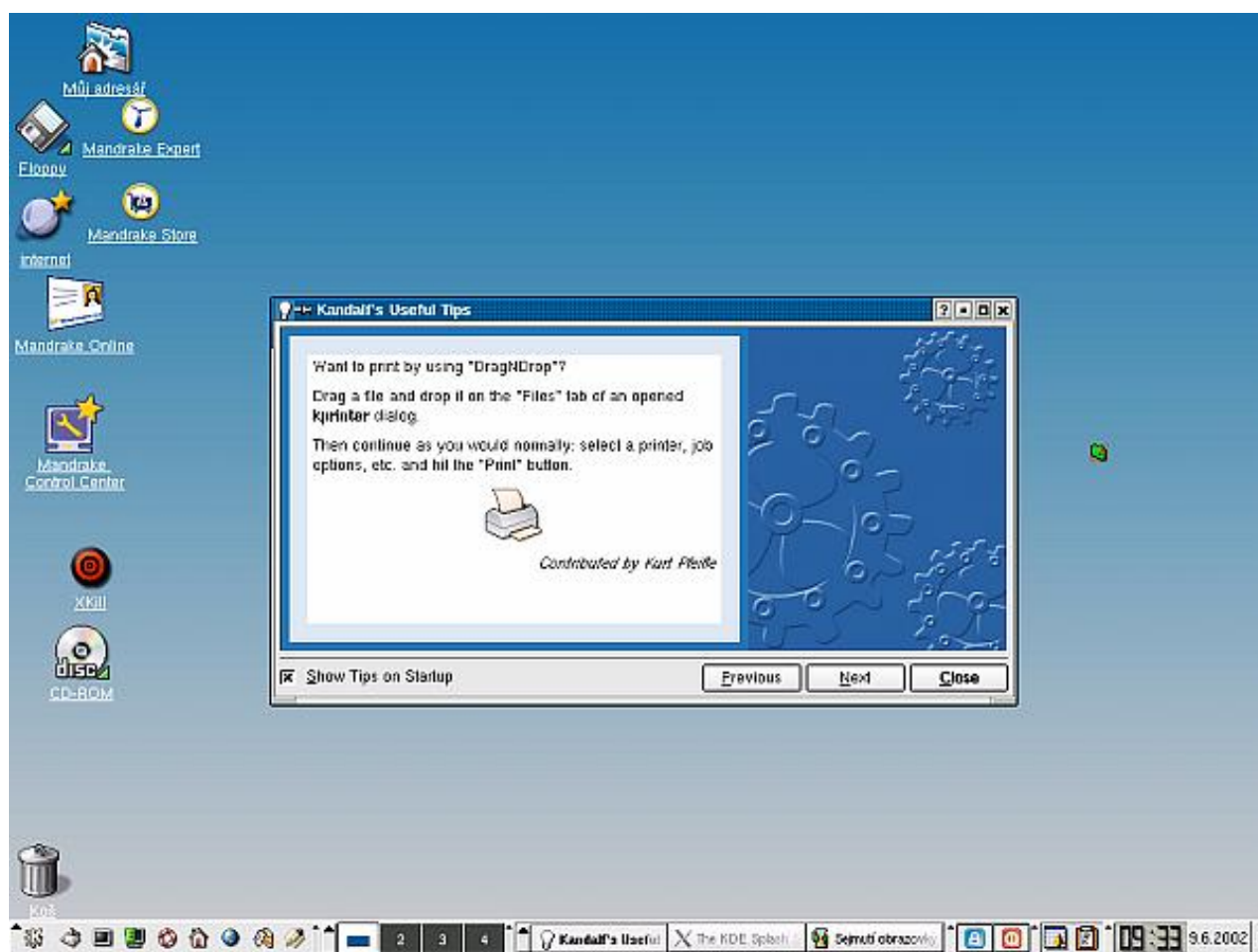
- Ü **/bin** – jsou zde uloženy základní příkazy k bootování, ke správě systému, změně hesel atd.
- Ü **/sbin** – podobně jako /bin, ale programy slouží jen superuživateli (rootovi), za jistých okolností je můžou používat i ostatní uživatelé.
- Ü **/boot** – soubory, jež používá zavaděč operačního systému, je zde uloženo jádro operačního systému LINUX, 300 – 700 KB ve zdrojovém kódu, po instalaci je nutná kompilace.
- Ü **/dev** – devices = adresáře zastupující hardware zařízení, jsou zde ke každému zařízení adresáře např. COA = COM1, cokoli sem zapíšeme, pošle se na COM1, NULL adresář – cokoli sem zapíšeme, se nestane.
- Ü **/etc** – databáze uživatelů, hesla, shelly atd., jsou zde uloženy konfigurační soubory programů (něco jako .ini v OS Microsoft Windows).
 /etc/rc, /etc/rc.d, /etc/rc?.d – jedná se o skripty a adresáře skriptů, které se spouští při startu, nebo v případě kdy se mění běh úrovně systému (init).
 /etc/passwd – databáze uživatelů systému s položkami, v nichž je uloženo uživatelské jméno, skutečné jméno uživatele, domovský adresář, šifrované heslo a další dodatečné informace.
- Ü **/home** – obsahuje domovské adresáře všech uživatelů (např: /home/novak), díky oddělení se snáze zálohují, často je tento adresář umístěn na jiném pevném disku.

- Ü **/lib** – sdílené knihovny pro různé programy.
/lib/modules – zaváděcí moduly jádra systému – zvláště ty, které jsou potřeba pro zavedení systému při zotavení po neočekávaných událostech (např. síťové ovladače a ovladače pro souborový systém).
- Ü **/mnt** – přípojný místo pro dočasná připojení dalších disků, například /mnt/cdrom).
- Ü **/proc** – tento adresář je virtuálně simulovaný, jsou zde ke každému procesu adresáře a je tam o tom procesu záznam, např. co se vykonávalo, kdo to přikázal, kdy, atd., při prohlížení těchto souborů jsou generovány operačním systémem.
- Ü **/root** – domovský adresář roota, je zvlášť kvůli bezpečnosti, v praxi ho často představuje jiný pevný disk.
- Ü **/tmp** – adresář vyhrazený pro pracovní soubory všeho druhu.
- Ü **/usr** - obsahuje všechny příkazy, knihovny, manuálové stránky a jiné soubory, jejichž obsah se nemění a které uživatel potřebuje při běžném provozu.
/usr/bin – veřejně přístupné příkazy,
/usr/doc – dokumentace,
/usr/lib – knihovny,
/usr/local – konfigurační soubory,
/usr/man – manuálové stránky tj. nápověda pro programy,
/usr/tmp – dočasné soubory programů,
/usr/X11R6 – X-window 11 release 6.
- Ü **/var** – proměnné – obsahuje soubory, které jsou časově závislé, především sdílené adresáře pro elektronickou poštu, systém -newsio, tiskárny, log-soubory, formátované manuálové stránky a dočasné soubory.
- Ü **Speciální adresáře** – Linux poskytuje dvě speciální jména adresářů.
Tečka (.) je stejná jako uvedení cesty k pracovnímu adresáři.
Dvě tečky (..) v cestě způsobí přechod o jednu úroveň zpět k nadřazenému adresáři.

5. Uživatelské rozhraní

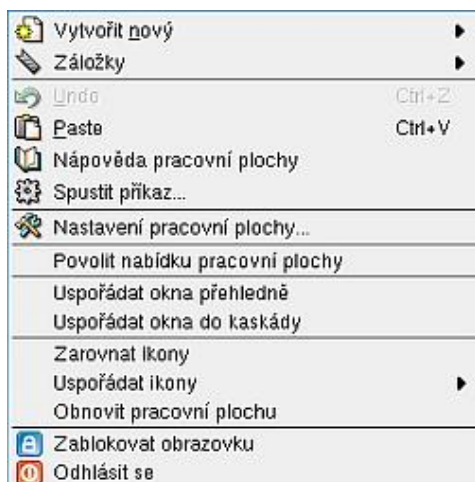
Po zapnutí počítače a nastartování Linuxu se na obrazovce objeví přihlašovací obrazovka. Pokud uživatel při instalaci své distribuce zvolil možnost Spouštět X při startu, má možnost nyní vidět přímo grafickou přihlašovací obrazovku. Nyní stačí již jen kliknout na ikonku u jména, do příslušného řádku vyplnit heslo a kliknout na tlačítko Přihlásit. Pokud se na obrazovce objeví pouze textová výzva, je nutné napsat přihlašovací jméno, stisknout Enter a do druhého řádku zadat heslo.

Pokud bylo vše zadáno správně, zobrazí se příkazový řádek, odkud je možné Linux ovládat. Pro zcela začátečnického uživatele postačí zatím pouze jediný příkaz: *startx*, popř. *startkde*. Po chvíli startování se zobrazí základní obrazovka systému:



Obr. 21 – Základní obrazovka systému

Pracovní plocha je velmi podobná ploše z operačního systému MS Windows – na ploše jsou ikonky, ve spodní části je ovládací panel. Okno zobrazené uprostřed obrazovky je okno uvítací a při každém spuštění vás obšťastní nějakým užitečným tipem. Pokud je uživatel opravdový začátečník, je tato pravidelně spouštěná nápověda dobrým způsobem, jak se něco naučit. Spouštění však lze samozřejmě i zakázat – stačí zrušit zatržení u položky Show Tips on Startup v levém spodním rohu okna.



Obr. 22 – Kontextové menu

Nastavení vzhledu plochy a ikon se provede podobně, jako u OS MS Windows. Stačí kliknout pravým tlačítkem na pracovní plochu a z kontextového menu zvolit položku Nastavení pracovní plochy. Poté se zobrazí okno s nastaveními, ve kterém je možné upravit vzhled plochy – vzhled ikon a písma na ploše, ale také barvu, prolínání barev, případně fotografii an pozadí plochy (tapetu). Dále je také možné nastavit šetřič obrazovky a jeho případnou vazbu na heslo.

Jedním z nejdůležitějších prvků celého systému je ovládací panel, který je možné rozdělit několik oblastí, které si stručně popíšeme. Těchto samostatných oblastí je šest a zleva následují v tomto pořadí:

První část obsahuje zástupce pro rychlé spuštění programů. Odstranit zástupce lze jeho přetažením do odpadkového koše. Přidání nového zástupce se provede přetažením jeho ikony přímo na panel. První ikona zleva, jakési ozubené kolo pod písmenem K, je ekvivalentem tlačítka Start známého z operačního systému MS Windows.



Obr. 23 – Panel zástupců

Druhá část ovládacího panelu obsahuje čtyři čtverce, které jsou zmenšeninami jednotlivých pracovních ploch. Je možné na nich vidět siluety jednotlivých otevřených oken. Přepnutí na vybranou plochu se provede kliknutím na její miniaturu.



Obr. 24 – Panel pracovních ploch

Ve třetí, největší části panelu se zobrazují náhledy jednotlivých oken. Minimalizovaná okna se schovávají právě do této části panelu a kliknutím na jejich název se znovu obnoví.



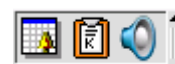
Obr. 25 – Panel minimalizovaných oken

Čtvrtá část panelu obsahuje pouze dvě tlačítka. Pomocí modrého tlačítka lze zamknout počítač v případě, že uživatel na nějakou dobu odchází. Oranžovým tlačítkem se provede odhlášení a vypnutí nebo restartování počítače.



Obr. 26 – Panel zamknutí a vypnutí

V předposlední části panelu se zobrazují stavové ikony zvláštních aplikací, tzv. appletů. Pokud se na ikonu klikne pravým tlačítkem, je možné zvolit její funkci nebo ji z panelu odebrat. Tyto ikonky lze přirovnat ke stavovým ikonám zobrazovaným v operačním systému MS Windows vedle hodin.



Obr. 27 – Panel stavových ikon

Poslední část panelu zobrazuje hodiny, které lze kliknutím pravého tlačítka myši nastavit k obrazu svému. Kliknutím levým tlačítkem se zobrazí kalendář – přehled aktuálního měsíce



Obr. 28 – Panel hodin

6. Systém ovládání souborů

6.1. Soubory

Pro UNIX je soubor posloupnost slabik bez jakékoliv další struktury. Rozlišujeme tři hlavní skupiny souborů:

- Obyčejné soubory
- Adresáře
- Speciální soubory

Každý soubor je ve vnitřní struktuře systému souborů reprezentován **i-uzlem** (i-node, index node). I-uzel obsahuje všechny atributy souboru a odkazy na datovou část souboru. Atributem souboru je např. informace, zda je soubor adresářem nebo obyčejným či speciálním souborem, kdo je jeho vlastníkem, jaká přístupová práva pro čtení nebo zápis mají uživatelé k souboru.

Uživatelé jsou rozděleni do skupin. Příslušnost ke skupině určí správce systému obvykle při registraci uživatele v systému. Uživatelé stejné skupiny jsou si potom bližší z hlediska přístupu k souborům než uživatelé jiných skupin.

6.2. Adresáře

Adresář je binární soubor s atributem příznaku adresáře a obsahuje seznam souborů. Po přihlášení je uživateli nastaven určitý adresář, který je označován jako domovský (home directory). Uživatel v průběhu sezení může měnit nastavení na různé adresáře. Adresář, který má právě nastaven, je označován jako pracovní (current directory). Obsah adresáře je zobrazitelný příkazem **ls**, ale ve skutečnosti jde o rozpis obsahu i-uzlů.

Hierarchie je završena adresářem označovaným "/", nazývaným také jako **kořenový adresář** (root directory). Kořenový adresář obsahuje podadresáře, které dále mohou obsahovat podadresáře do teoreticky libovolné hloubky. Celý systém souborů je spojen ve strom adresářů. V rámci tohoto stromu má uživatel přidělen adresář, který je mu nastaven po přihlášení do systému a kterým začíná jeho uživatelská oblast dat (jeho uživatelský podstrom). Je to jeho domovský adresář (home directory).

Výpis jména pracovního adresáře získáme příkazem *pwd*.

Adresář obsahuje obyčejné soubory. Základní manipulace se soubory jsou kopie, přejmenování a zrušení souboru.

6.3. Speciální soubory

Jména speciálních souborů se řídí konvencemi danými výrobcem, nikoliv obecnou normou. Přesto existují zvyklosti pro jména speciálních souborů jako např.:

<i>console</i>	operátorská konzola
<i>clock</i>	hodiny
<i>fd</i>	disketa
<i>dsk</i>	disk
<i>kmem</i>	operační paměť jádra
<i>lp</i>	tiskárna
<i>mem</i>	operační paměť
<i>mt</i>	klasická magnetická páska
<i>tty</i>	terminál

Uvedené texty zastupují pouze používaný základ jména, protože např. disketa pro sekvenční přístup (po znacích) mívá předponu "r", tj. "rfd" (stejně tak mg. páska) a dále u více připojených periférií stejného typu je příponou jména pořadí periferie, např. "tty00", "tty01", ...

Přístup k periférii je buď znakový nebo blokový. Typicky blokové zařízení je disk, znakové terminál. Ale i s diskem můžeme pracovat sekvenčně po znacích, proto je i pro něj vytvořen znakový speciální soubor. Znakový speciální soubor má při výpisu "ls -l" atributů v prvním sloupci znak 'c', blokový znak 'b'.

6.4. Práva k souborům

Pomocí příkazu *chown* můžeme měnit vlastníka souboru. Příkazem *\$ chown root soubor* předáme vlastnictví souboru "*soubor*" uživateli "*root*". Příkaz *chmod* mění přístupová práva souboru. Změnu můžeme zapsat buďto číselně, např. *\$ chmod 775 davka*, což je specifikace v osmičkové soustavě přístupových práv. Souboru se jménem "*davka*" budou přístupová práva nastavena způsobem:

```

rwxrwxr-x
11111101
7 7 5

```

To odpovídá hodnotě v osmičkové soustavě jednotlivých cifer vždy pro trojici bitů oprávnění. Mnemonický zápis vychází ze zadání vlastníka souboru (*u=user*), skupiny (*g=group*), ostatních (*o=others*) nebo všech (*a=all*) uživatelů, a označení přístupových práv "*r*" (*r=read*) pro čtení, "*w*" (*w=write*) pro zápis a "*x*" (*x=execute*) pro provádění.

6.5. Tabulka datových bloků

Tabulka datových bloků v i-uzlu má následující organizaci. Prvních 10 hodnot tabulky (0-9) odkazuje přímo na datové bloky. Jedenáctá hodnota je odkaz na blok nepřímé reference, což znamená, že teprve odkazovaný blok obsahuje vlastní odkazy na datové bloky. Je-li vyčerpán, dvanáctá hodnota je odkaz na blok druhé nepřímé reference (odkaz na blok obsahující odkazy na bloky teprve s referencí na data) a je-li i tato možnost vyčerpána, třináctá hodnota je odkaz na blok třetí nepřímé reference

7. Systém správy paměti

7.1. Jádro

Jádro používá stránkování k vytvoření virtuální paměti. Každý proces má svůj oddělený adresní prostor a tak si programy nevidí navzájem „do karet“ a nemůžou se navzájem ohrozit. Také nevidí do paměti jádra a tak nemohou ohrozit ani to. Jádro dovoluje procesu jeho adresní prostor modifikovat. Umí „namapovat“ soubor na danou pozici – v paměti je pak vidět obsah daného souboru, který jádro v případě potřeby nahraje, nebo obnoví na disku (pokud program paměť modifikoval). Sdílet určitou část paměti mezi procesy apod.

Standardně se paměť procesu skládá z jeho kódu, který program nemůže měnit, neomezeně rostoucího zásobníku a z druhé strany adresního prostoru rostoucí haldy.

Navíc jádro vytváří virtuální paměť a odkládá části na disk, pokud to je třeba.

Jádro Linuxu obsahuje velmi kvalitní správu paměti. Mezi jeho přednosti patří:

- Ü Disková cache, která se dynamicky mění podle velikosti volné paměti.
- Ü Zavádění programu „po kouskách“ – při startu programu se jeho kód nenahrává celý do paměti. Nahraje se pouze jeho první stránka a program se spustí. Pokud potřebuje další stránky, tak se mu nahrají později. Toto zrychluje start programů a šetří paměť.
- Ü Zpožděné přidělování paměti – pokud program požádá o volnou paměť, jádro si to pouze zapamatuje a přidělí ji až v době, kdy ji program začne používat.
- Ü Zpožděné kopírování paměti – když se běžící proces rozdělí na dva, jeho paměť by se měla zkopírovat. To se ale nestane a kopie se vytváří se zpožděním až v situaci, když jeden z programů začne do ni zapisovat.

7.2. Alokace paměti v jádře

Linux i FreeBSD byly navrženy tak, že stránky alokované jádrem jsou neswapovatelné. Struktury jádra zabírají pouze malou část paměti, a proto jejich neswapovatelnost nevadí. Existují i systémy, které mají některé struktury jádra swapovatelné - například AIX nebo Windows NT - nicméně to komplikuje programování v jádře a přináší to s sebou problémy. Proces se může zablokovat kdykoli při přístupu na swapovatelnou paměť, proto je nutné v tomto místě počítat s možným přepnutím procesu. Není tedy možno držet spinlock a přistupovat na swapovatelnou paměť.

Základní funkcí systému je operace alokace stránky. Každá fyzická stránka paměti je popsána strukturou *struct page* na Linuxu a *struct vm_page* na FreeBSD. Pole těchto struktur je staticky alokováno při startu systému. Jeho velikost je určená podle množství paměti. Požadavky na stránky se v zásadě dělí na dvě skupiny – na blokuující a neblokuující (též zvané atomické) alokace paměti. Při blokujícím požadavku je možné, že proces se zablokuje a počká, než swapper nějaké stránky odswapuje na disk nebo než uvolní stránky z cache nebo diskových bufferů. Při atomickém požadavku se proces zablokovat nesmí, a pokud dojde paměť, alokační funkce vrátí *NULL*. Protože systém za běhu používá skoro celou paměť pro uživatelské procesy a jako diskovou cache, je v něm za běhu poměrně málo volných stránek a k selhání atomické alokace může dojít v podstatě kdykoli. Platí obecný princip: při blokující alokaci je garantováno, že proces paměť dostane (na Linuxu se může stát, že blokující alokace vrátí *NULL*, ale dochází k tomu jen

v extrémním případě, kdy jádro zabralo celou paměť; na FreeBSD blokující alokace neselže nikdy, a pokud jádro zabere celou paměť, je zastaveno na *panic*). Naproti tomu atomická alokace může kdykoli selhat a kód se s tím musí vypořádat bez nějaké ztráty dat nebo narušení funkčnosti. Atomická alokace se používá v přerušení, neboť obslužná rutina přerušení se zablokovat nesmí. Typickým použitím atomické alokace je alokace paměti pro pakety přicházející ze sítě. To se dělá v přerušení od ovladače síťové karty a selhání atomické alokace zde nevadí, neboť protokoly vyšších vrstev se se ztrátou packetu musejí umět vypořádat.

K zajištění dobré funkčnosti systému je třeba mít nějaký limit pro atomické alokace. Pokud je množství volných stránek menší než tento limit, je paměť přidělována už jen atomickým alokacím a blokující alokace musejí počkat, než se stránky odswapují a volná paměť stoupne nad limit. Kdyby takový limit neexistoval, mohlo by se například stát, že proces neustále alokující stránky by zablokoval celé síťování, neboť by okamžitě zabral jakoukoli uvolněnou stránku, na obslužnou rutinu přerušení síťové karty by už nezbylo nic a každý packet by byl zahozen.

7.3. Buddy alokátor na Linuxu

Na Linuxu slouží k alokaci stránek funkce `struct page *alloc_pages(unsigned int gfp_mask, unsigned int order)`. První parametr je konstanta `GFP_xxx` nebo maska složená z konstant `__GFP_xxx`. `GFP_ATOMIC` určuje, že jde o atomickou alokaci, `GFP_KERNEL` znamená, že jde o běžnou blokující alokaci. `GFP_USER` se používá pro alokaci stránek, o které požádají uživatelské procesy – má stejný význam jako `GFP_KERNEL` až na to, že pokud je zaplněná celá paměť i swap, je limit pro `GFP_USER` vyšší než pro `GFP_KERNEL`. Je to proto, aby, když uživatelské procesy spotřebují celou paměť, ještě nějaká paměť zbyla jádru. `GFP_NOFS` a `GFP_NOIO` znamená, že alokátor nesmí zavolat filesystém nebo blokový io-systém, aby uvolnil paměť, protože by mohlo dojít k deadlocku. Použití těchto dvou příznaků mi nepřipadá moc čisté – alokátor sice nemůže zavolat filesystém nebo io-systém, nicméně může klidně čekat, než swapovací proces `kswapd` nějakou paměť uvolní – a `kswapd` volá filesystém i io-systém. Jak uvidíme později, ono je to s těmi deadlocky docela zamotané a ne moc dobře vyřešené. Další příznaky určují, zda paměť musí ležet v dolních 16M paměti, aby byla použitelná pro ISA DMA, nebo zda může ležet v nenamapované high-memory zóně, o které bude řeč později.

Druhý parametr funkce `alloc_pages` je dvojkový logaritmus počtu stránek, který se má alokovat. Je-li to nula, alokuje se jedna stránka, je-li to 1 či 2, alokují se dvě či čtyři stránky a tak dále. Alokované stránky jsou v jednom souvislém bloku. Linux používá k alokaci buddy alokátor, který drží pro jednotlivé mocniny dvojky seznamy bloků příslušného počtu volných stránek. Je držen seznam všech volných nespárovatelných stránek. Dále je držen seznam všech volných dvojic stránek (dvojice stránek musí být zarovnaná – t.j. dolní stránka má pořadí, které je dělitelné dvěma), pak je držen seznam všech volných čtveřic stránek (dolní stránka čtveřice má pořadí dělitelné čtyřmi) a tak dále i pro vyšší mocniny dvojky. Při požadavku o alokaci určitého počtu stránek se alokátor nejdříve podívá do seznamu odpovídajícího danému počtu. Pokud tam blok stránek nalezne, tak ho přidělí. Pokud ho tam nenalezne, podívá se do dalšího seznamu (obsahujícího bloky dvojnásobné velikosti), a když tam nalezne blok, rozdělí ho na poloviny. Jednu polovinu uloží do nižšího seznamu bloků poloviční délky a druhou polovinu vrátí. Pokud nenalezne blok ani tam, podívá se do dalšího seznamu a tak dále, až narazí na seznam největších bloků. Pokud nenalezne volný blok ani tam, alokace selže. Při uvolnění bloku stránek se kontroluje, zda onen blok nemá volný blok v páru, a pokud ano, oba bloky se spojí a přidají se do vyššího seznamu. Pokud nový spojený blok má opět volný blok v páru, spojí se i tyhle, a tak dále.

Buddy alokátor s sebou přináší problém – a tím je fragmentace paměti. Pokud není spotřebovaná celá paměť, je garantováno, že se podaří alokovat jednu stránku; bohužel už není garantováno, že se podaří alokovat více souvislých stránek. Na starších jádrech 2.4 platilo, že

alokace mohla kdykoli selhat, pokud parametr order byl větší než nula. Alokace dvou souvislých stránek je na IA32 potřeba k vytvoření procesu, proto se tam mohlo kdykoli stát, že vytvoření procesu selže a vrátí chybu ENOMEM. V jádře 2.4.18 je to napsáno tak, že pokud je $\text{order} \leq 3$ a alokace selže (a nejedná se o atomickou alokaci), bude se v cyklu neustále volat funkce na uvolnění nebo swapování stránek, doufajíc, že se blok potřebné velikosti podaří vytvořit (v dřívějších jádrech tento cyklus probíhal jen pro $\text{order} == 0$). Nepovede to sice k náhodnému selhání vytváření procesu jako dřív, nicméně při velmi nevhodném umístění stránek a nedostatku swapu může dojít k nekonečnému cyklu. Například pokud je alokována polovina paměti a stránky jsou tak nešikovně umístěny, že volné stránky netvoří žádný pár, nepodaří se vytvořit nový proces a tahle operace bude cyklit v nekonečné smyčce. Nicméně pravděpodobnost, že se to stane, je velmi malá. Pokud je navíc dostatek swapovacího místa, budou stránky swapovány tak dlouho, dokud se blok potřebné velikosti nevytvoří.

7.4. Barvení stránek na FreeBSD

FreeBSD nemá buddy alokátor. K alokaci jedné stránky se používá funkce `struct *vm_page *vm_page_alloc(vm_object_t object, vm_pindex_t index, int page_req)`. První parametr je objekt, do kterého se má stránka uložit, a druhý je index v tomto objektu (vm_objekty budou popsány později; pro alokaci stránek pro jádro se používá objekt jádra `kernel_object`). Třetí parametr určuje prioritu alokace - nejnižší pro uživatelský proces, vyšší pro systém, nejvyšší pro interrupt. Tato funkce se nikdy neblokuje. Pokud paměť není nebo pokud je množství volné paměti pod kvótou pro danou prioritu alokace, vrátí NULL. Pokud vrátí NULL, může se kód, který ji volal, zablokovat pomocí makra `VM_WAIT`; a čekat, než swapper nějaké stránky odswapuje a nějaká paměť bude dostupná. Po probuzení z `VM_WAIT` je třeba opět zkusit alokaci pomocí `vm_page_alloc`.

Při alokaci používá FreeBSD algoritmus barvení stránek. Barvení stránek slouží k optimalizaci použití L2 cache. Pro pochopení algoritmu nejdříve popíši, jakým způsobem cache fungují. Nejmenší jednotka, s jakou cache pracuje, je řádka. Například na Pentiu až Pentiu 3 má řádka velikost 32 bytů, na Pentiu 4 má velikost 64 bytů. Ačkoli se říká, že cache je asociativní paměť, není to pravda. Udělat plně asociativní cache není technicky možné, proto má cache adresy. Dolních 5 nebo 6 bitů v adrese je pozice uvnitř řádky, dalších několik bitů adresy je adresa v cachi. Cache pracuje jako obyčejná paměť - těchto několik bitů se použije jako paměťová adresa a podle ní se v cachi najde příslušné místo, na kterém je uloženo několik řádek (typicky 1, 2 nebo 4 – podle toho se cache nazývá jednocestně, dvoucestně nebo čtyřcestně asociativní). Požadovaná celá adresa se porovná s celými adresami těchto několika řádek, a pokud je jedna z nich shodná, prohlásí se hodnota v cachi za nalezenou. Například Pentium 2 má L1 cache čtyřcestně asociativní o velikosti 16kB s délkou řádky 32B. Znamená to tedy, že dolních 5 bitů adresy je offset uvnitř řádky. Dalších 7 bitů je adresa v cachi (spočteno jako $\log_2(16\text{kB}/32\text{B}/4)$). Na jedné adrese v cachi se nacházejí čtyři řádky. U Pentia 2 je L2 cache čtyřcestně asociativní, má velikost 512kB a délku řádky 32B. To znamená, že 5 bitů adresy je offset v řádce a 12 dalších bitů je adresa v cachi.

Cache pracuje s fyzickými adresami. (Existují i architektury – například Sparc64 – kde cache pracuje s virtuálními adresami. Je s tím velké množství problémů, pokud dvě virtuální stránky ukazují na jednu stránku fyzickou.) Pokud je například cache čtyřcestně asociativní a uživatelský program alokuje pět souvislých po sobě jdoucích stránek, může se stát, že stránky náhodou padnou na takové fyzické adresy, že všech pět stránek bude mít stejnou adresu v cachi. Pokud pak program bude tyto stránky procházet, nepodaří se je do cache dostat, neboť na jedné adrese čtyřcestně cache mohou být pouze čtyři řádky. I když má L2 cache značnou velikost 512kB, je možné, že se pět nevhodně umístěných stránek o celkové velikosti 20kB do téhle cache nevejde.

Aby k tomuto jevu nedocházelo, používá FreeBSD barvení stránek. Každá stránka má barvu, což je adresa dat v cachi. Například pokud máme čtyřcestnou cache o velikosti 512kB a velikost stránky 4kB, stránky mají 32 barev. Alokátor stránek ve FreeBSD se snaží přidělovat stránky takové barvy, která odpovídá virtuální adrese, na níž bude stránka namapována. Pokud například program alokuje pět stránek na virtuálních adresách 0, 4096, 8192, 12288 a 16384, budou tyto stránky mít barvy 0, 1, 2, 3, 4. Touto strategií se zabrání výše zmiňovanému jevu, kdy několik stránek má stejnou barvu (čili stejnou adresu v cachi) a tyto stránky se pak do cache nevejdou. Implementace barvení stránek je jednoduchá – pro každou barvu existuje fronta volných stránek, které onu barvu mají. Při požadavku o stránku se nejprve bere stránka z fronty s požadovanou barvou. Pokud je fronta prázdná, vezme se stránka z fronty s nejvzdálenější barvou. Je-li tato fronta také prázdná, procházejí se ostatní fronty. Parametr index funkce `vm_page_alloc` se používá k určení požadované barvy stránky.

Otázka, zda barvení stránek pomáhá, nebo nepomáhá zvýšit rychlost běhu programů, je sporná. Tvůrci Linuxu tvrdí, že barvení zbytečně zesložitňuje jádro a jeho efekt je zanedbatelný, a proto ho nechtějí implementovat. Protože tvůrci FreeBSD barvení implementovali, tvrdí, že barvení pomáhá při zvýšení rychlosti. Realita je asi taková, že:

- Ü Pokud program alokuje několikrát méně paměti, než je velikost L2 cache, pak pokud barvení stránek nebude použito, je malá pravděpodobnost, že bude mít spousta stránek stejnou barvu. Proto v takovém případě nemá barvení velký vliv.
- Ü Pokud program alokuje několikrát více paměti, než je velikost L2 cache, a náhodně na tuto paměť přistupuje, tak se stránky do cache stejně nevejdou a vůbec nezáleží na tom, zda je barvení použito, nebo ne.
- Ü Pokud program alokuje množství paměti srovnatelné s velikostí L2 cache, má barvení stránek význam. Náhodným přidělováním stránek bez barvení se nedosáhne rovnoměrného rozdělení barev a pokrytí celé cache. Naproti tomu barvení zajistí, že cache bude rovnoměrně pokrytá a že program dostane přesně takové množství stránek se stejnou barvou, jaká je asociativita cache. Díky tomu se celá datová oblast programu vejde do cache

8. Správa procesů

8.1. Scheduler

Správa procesů na Linuxu, FreeBSD a ostatních Unixových systémech se provádí pomocí tzv. Scheduleru. Scheduler je plánovač procesů - rozhoduje, který proces na kterém procesoru poběží. Na přepínání procesů jsou kladeny požadavky, do určité míry protichůdné:

- Ü Nesmí docházet k příliš častému přepínání procesů. Přepnutí procesu je poměrně náročná operace, která nějakou dobu trvá, proto je nežádoucí, aby procesor trávil mnoho času přepínáním procesů místo vykonávání užitečného kódu.
- Ü Nesmí docházet k málo častému přepínání procesů, pak by uživatel získal dojem, že systém reaguje pomalu.

Pokud nějaký proces čeká na diskové I/O, klávesu na terminálu, data ze sítě nebo jinou událost, musí být probuzen a spuštěn okamžitě, jakmile tato událost nastane. Kdyby například proces četl data z disku a byl spuštěn pozdě, vedlo by to ke značnému poklesu přenosové rychlosti disku.

Na víceprocesorových systémech je žádoucí, aby proces nestřídal procesory a běžel pokud možno pouze na jednom procesoru. Každý procesor má svoji vlastní cache. Pokud je proces přehozen na jiný procesor, budou jeho data přelévána z cache původního procesoru do cache nového procesoru. Toto přelévání po sběrnici je velmi pomalé.

Uživatel musí mít možnost nastavit prioritu procesu. U určitých procesů (typicky se jedná o nějaké dlouhodobé výpočty) je požadováno, aby běžely jen, když žádný jiný proces nepožaduje procesor.

Každý proces se může nacházet v několika stavech:

- Ü Proces je zablokován, pokud čeká na nějakou událost. Proces čeká na nějaké čekací frontě, semaforu nebo jiném synchronizačním primitivu. Takový proces není možno spustit. Zablkování máme dvojího druhu: přerušitelné signálem (procesy v tomto stavu jsou označeny písmenem "S" ve výpisech z příkazů ps nebo top) a nepřerušitelné signálem (jsou označeny písmenem "D"). Pokud proces dostane signál a je v přerušitelném čekání, aktuální syscall se zruší, proces se vrátí do userspace a provede se obsluha signálu. Pokud to bylo při registraci signálu požadováno (příznak SA_RESTART), pak po skončení obsluhy signálu libc zavolá znovu přerušovaný syscall. Nepřerušitelné zablkování se používá například při operacích s diskem. Při nich máme zaručeno, že operace skončí v krátkém čase, takže proces nebude příliš dlouho zablokován. Nepřerušitelné zablkování výrazně zjednodušuje implementaci filesystému. Přerušitelné zablkování se používá, pokud čekání může trvat hodně dlouho - při čtení z terminálu a při operacích s pipami a sockety. Scheduler mezi těmito dvěma způsoby zablkování nerozlišuje.
- Ü Proces je připraven, pokud je možno jej spustit, ale neběží (neboť běží jiný proces). Do připraveného stavu se dostane buď tak, že běží a je preemptivně přepnut, nebo pokud byl zablokován, probuzen, ale ještě nespuštěn.
- Ü Proces je ve stavu běžící, pokud je jeho kód právě na procesoru vykonáván. Pokud proces běží, může se buď zablokovat, nebo může být preemptivně přepnut, čímž se dostane do stavu připraven.

8.2. Scheduler na Linuxu 2.4 a nižších

Algoritmus plánování procesů na Linuxu je stejný již od první verze 0.01 až po 2.4. Každý proces je popsán velikou strukturou `task_struct`. Položka `state` této struktury určuje stav procesu - může nabývat hodnot `TASK_INTERRUPTIBLE` (přerušitelné zablokování), `TASK_UNINTERRUPTIBLE` (nepřerušitelné zablokování) a `TASK_RUNNING` (proces je připraven nebo běží). Existují dva obousměrně linkované seznamy procesů - seznam všech procesů (položky `next_task` a `prev_task`) a seznam procesů ve stavu `TASK_RUNNING` (položky `next_run` a `prev_run`). Pro účely scheduleru má `task_struct` položky `nice` a `counter`. `nice` je priorita, kterou uživatel nastavil pomocí systémového příkazu `nice`. V příkazu `nice` se nastavuje číslo v rozsahu -20 (nejvyšší priorita) až 19 (nejnižší priorita); pro potřeby scheduleru je toto přepočítáno na rozsah 1 (nejnižší priorita) až 40 (nejvyšší priorita). `counter` je počet tiků, které procesu ještě zbývají. Při vzniku procesu se `nice` zkopíruje do `counter`.

Jádro scheduleru se nachází ve funkci `void schedule(void)`. Tato funkce odstaví aktuální proces ze seznamu připravených procesů, pokud jeho stav není `TASK_RUNNING`. Pak vybere ze seznamu připravených procesů proces s největší hodnotou `counter` a ten spustí. Při běhu procesu se `counter` zmenšuje o jedničku při každém tiku. Proces běží, dokud `counter` nedosáhne nuly nebo dokud se sám nezablokuje. Pak se opět zavolá `schedule` a ta vybere další proces. Pokud funkce `schedule` zjistí, že všechny připravené procesy mají `counter` roven nule, provede pro všechny (připravené i zablokované) procesy operaci $\text{counter} = \text{counter} / 2 + \text{nice}$. Pro připravené procesy tato operace pouze přiřadí `nice` do `counter`, pro zablokované procesy způsobí, že `counter` bude pomalu růst až k hodnotě $2 * \text{nice}$. Tím je zajištěno, že proces, který je delší dobu zablokovaný, bude mít vyšší prioritu než všechny běžící procesy a po probuzení bude spuštěn okamžitě. Na starších jádrech byla hodnota `nice` používána rovnou (což znamenalo, že při tiku 100Hz dostal proces s implicitní prioritou časové kvantum 200ms), v novějších 2.4 je lineárně přepočítávána tak, aby proces s implicitní prioritou měl přibližně 50ms časové kvantum. Ač je tento algoritmus velmi jednoduchý, splňuje většinu požadavků na scheduler kladených: pokud běží několik procesů, střídají se a dostávají časová kvanta úměrná jejich prioritě. Pokud je nějaký proces delší dobu zablokovaný, naroste mu `counter` na větší hodnotu než všem ostatním procesům, je po probuzení spuštěn okamžitě a má až dvakrát větší časové kvantum. Pokud je proces zablokovaný jen na velmi krátkou dobu, `counter` mu příliš nenaroste.

Horší je to už s víceprocesorovými systémy. Byla snaha zabránit přehazování procesů mezi procesory tak, že při výběru procesu s největším `counter` je k této hodnotě přičteno 5, pokud proces běžel naposledy na procesoru, na kterém se schedulování provádí. Rovněž je přičteno 1, pokud proces má stejnou virtuální paměť jako proces, který na procesoru právě běžel - neboli pokud se jedná o thready. Přepínání mezi thready je méně náročné než přepínání mezi procesy. Toto nám zajišťuje, že pokud budeme mít dvouprocesorový systém a na něm nám poběží sudé množství procesů, bude mít každý proces svůj procesor a procesy nebudou mezi procesory přehazovány. Nicméně pokud nám na dvouprocesorovém systému poběží tři procesy, tak k cyklickému přehazování jednoho nebo všech procesů mezi procesory dojde. Optimální strategie v takovém případě by byla nechat na jednom procesoru běžet dva procesy, na druhém procesoru jeden proces a přehození procesu provádět pouze za dlouhou dobu, aby všechny procesy dostaly stejné procento výpočetní síly.

Další nevýhodou tohoto scheduleru je jeho značná časová složitost. Každé přepnutí procesu má složitost lineární vzhledem k počtu připravených procesů. To není ještě tak zlé - připravených procesů je v systému jen pár; systém s více než dvaceti připravenými procesy je stejně nepoužitelný (neboť tam každý proces dostane pouze dvacetinu výpočetního času), takže uživatelé tak velké zátěže na systém nedávají. Nicméně pokud všem dojde `counter` na 0, musí se projít všechny procesy v systému, a to je značně náročná operace, neboť procesů může být až několik

stovek. Složitost jednoho přepnutí procesu je tedy $O(\text{počet připravených procesů} + \text{počet všech procesů} / \text{počet připravených procesů})$.

Linux má reálné procesy. Pokud v systému existuje nějaký připravený reálný proces, je spuštěn přednostně před všemi ostatními procesy. K nastavení reálné priority procesu je potřeba právo superuživatele, neboť tyto procesy jsou poměrně nebezpečné; pokud se v reálném procesu vyskytne nekonečná smyčka, celý systém zatuhne. Reálné procesy mají uživatelsky nastavené priority, spuštěn je vždy reálný proces s nejvyšší prioritou, pokud má více reálných procesů stejnou prioritu, cyklicky se mezi nimi přepíná. Reálné procesy nejsou reálné v pravém slova smyslu - je sice garantováno, že reálný proces bude spuštěn před všemi ostatními procesy, nicméně doba spuštění garantována není. Pokud například nějaký nereálný proces stráví příliš mnoho času v jádře, aniž by se zablokoval nebo zavolal funkci podmíněného přepnutí `cond_resched`, tak reálný proces spuštěn být nemůže. K vylepšení doby odezvy je třeba použít low-latency patch nebo preemptivní přepínání procesů uvnitř jádra.

8.3. Scheduler na Linuxu 2.5 a 2.6

V Linuxu 2.5 byl scheduler zcela přepsán, aby měl složitost $O(1)$ a aby lépe fungoval na SMP. Scheduler pracuje zvlášť na každém procesoru a má frontu zvlášť pro každý procesor. Každý proces má proměnnou `sleep_avg`, která se zvětšuje o 1 každý tik, kdy proces spí, a zmenšuje se o 1 každý tik, kdy proces běží. Má určitou maximální velikost, kterou nesmí překročit. Efektivní priorita procesu je určena jako statická priorita nastavená uživatelem \pm bonus, kde bonus je `sleep_avg` lineárně zobrazený do intervalu -5 - +5. Připravené a běžící procesy jsou uloženy ve frontách příslušejících jejich efektivním prioritám. K frontám existuje bitová maska neprázdných front, aby se mohla snadno najít neprázdna fronta s nejvyšší prioritou. Pro každou prioritu existují dvě fronty: aktivní a záložní. Každý proces dostane `time_slice`, což je uživatelem nastavená priorita lineárně zobrazená do intervalu 10ms -- 300ms. Systém vybere z aktivních front proces s nejvyšší efektivní prioritou a pustí ho. Proces běží a ubírá svůj `time_slice`. Až `time_slice` dojde, proces je přeřazen z aktivní fronty do záložní fronty příslušející jeho prioritě. Pak je vybrán další proces s nejvyšší prioritou z aktivních front. Když už v aktivních frontách žádné procesy nejsou, tak se záložní a aktivní fronty vymění a provede se znovu výběr procesu.

Pokud je úloha interaktivní (což je rozhodnuto lineárně na základě `sleep_avg` a uživatelské priority), je při ztrátě `time_slice` opět vložena do aktivní fronty. V takovém případě by mohlo docházet k hladovění procesů v záložní frontě - pokud k tomu dojde (což se pozná tak, že proces v záložní frontě s nejvyšší prioritou neběžel po určitou dobu), tak se interaktivní procesy budou dávat do záložní fronty jako obyčejné procesy.

Aby scheduler dobře fungoval na SMP, celý tento mechanismus funguje zvlášť na každém procesoru. Jednou za 250ms, nebo pokud je nějaký procesor volný, je prováděn load-balancing, při kterém se systém snaží vyrovnat velikost front. Procesor, na kterém se load-balancing provádí, najde nejvytíženější procesor. Pokud je tento procesor vytížen více jak na 4/3 zátěže procesoru provádějícího load-balancing, je mu nějaký proces odebrán.

Tento scheduler je určitě výrazně lepší než ve starších verzích Linuxu: všechny operace zde mají složitost $O(1)$; procházení seznamu všech procesů nebo seznamu všech připravených procesů se zde nedělá nikde. Rovněž je zajištěna dobrá funkce na SMP, neboť procesy nejsou svévolně přehazovány mezi procesory. Procesy jsou přehazovány jen při operaci load-balancingu.

8.4. Scheduler na FreeBSD 4 a nižších

Scheduler na FreeBSD funguje následovně – máme 32 front příslušejících prioritám. Scheduler vybírá procesy z fronty s nejvyšší prioritou a mezi nimi cyklicky přepíná. Každý proces běží 100ms. Každý proces má proměnnou `p_estcpu`. Tato proměnná roste lineárně, když proces běží, a klesá exponenciálně, když proces neběží. Rychlost klesání je ovlivněna systémovou zátěží (load average, vypsáno příkazem `uptime`). Za $5 \cdot \text{load_average}$ sekund `p_estcpu` klesne na 10%. Skutečná priorita je určena lineárně z uživatelské priority a `p_estcpu`. Scheduler jednou za sekundu prochází všechny procesy a snižuje `p_estcpu`, čímž zvyšuje prioritu neběžícím procesům. Tím je zajištěno, že procesy nehladoví. Pokud proces čeká na nějakou událost pomocí funkce `tsleep` nebo `asleep`, může v parametru funkce určit, s jakou prioritou bude probuzen. Scheduler přepíná procesy se složitostí $O(1)$, ovšem jednou za sekundu prochází všechny procesy.

Scheduler na starších FreeBSD 4 na SMP obsahoval naprosto neuvěřitelný bug, který způsobil, že pokud jsou všechny procesory obsazeny, mají interaktivní procesy až několikasekundovou odezvu. V současné FreeBSD 4.9 už naštěstí není.

8.5. Kernel schedulable entities na FreeBSD 5

Na FreeBSD 5 byl scheduler zcela přepsán, aby lépe umožňoval použití threadů. Nebyl změněn jen algoritmus, ale i význam datových struktur.

Existují dvě základní metody, jak implementovat thready:

- **Kernel thready** - thready přepínané v jádře. Jádro se ke threadům chová stejně jako k samostatným procesům, které mají společnou virtuální paměť, a přepíná je. Takto jsou thready implementovány na Linuxu a na starších FreeBSD.
- **User thready** - jádro vidí jeden proces, přepínání mezi thready provádí samotná threadová knihovna běžící v userspace. Takto funguje FreeBSD 4 nebo dnes již zapomenutá knihovna NSPR na Linuxu, kterou používal Netscape.

User thready mají nevýhodu, že pokud se jeden thread dostane do nepřerušitelného zablokování, žádné další thready nemohou běžet, protože nemůže být doručen časovací signál knihovně, která thready přepíná. Implementace přerušení přerušitelného čekání je značně problematická. Navíc user thready nemohou efektivně využívat víceprocesorový stroj, neboť pro jádro se celý multithreadový program jeví jako jeden proces, který je spuštěn na jednom procesoru.

Naproti tomu kernel thready mají tu nevýhodu, že pokud thready čekají na sebe navzájem, musí se přitom provádět volání jádra, což se při user threadech nemusí. User thready také narozdíl od kernel threadů nekonzumují žádné zdroje jádra.

Na FreeBSD byla snaha napsat hybridní thready, které by měly výhody jak kernel, tak user threadů. Idea je taková, že se thready budou chovat jako thready userspaceové, ale pokud se nějaký thread zablokuje v jádře, bude vyrobena nová struktura popisující kontext jádra a ostatní thready procesu poběží s touto novou strukturou, zatímco stará struktura zůstane zablokovaná v jádře. Proto byl zcela změněn význam některých struktur jádra a přibýly struktury nové. `struct proc` popisuje proces stejně jako v předchozích verzích. Nicméně tato struktura již neobsahuje informace pro scheduler a neobsahuje kontext jádra (t.j. zásobník jádra a příznaky související se zablokováním a odblokováním). Kontext jádra byl přesunut do struktury `struct thread`. Proces má několik struktur `struct kse` (kernel schedulable entity) - pro každý procesor jednu. Ty se z hlediska

schedulování chovají jako samostatné procesy. Každá tato struktura má jednu struct thread obsahující kontext jádra. Pokud se tento kontext zablokuje někde v jádře, je alokována nová struct thread, ta se vrátí do userspace ve speciálním upcallu a s ní daná KSE dále poběží (upcall provede výběr dalšího připraveného threadu a jeho spuštění). KSE jsou sdruženy v struct ksegrp, která obsahuje obecné informace pro scheduler, jako například p_estcpu. Uvnitř jednotlivých KSE pak běží samotné userspaceové thready, přepínané schedulerem v userspace. To zajistí výhody rychlého přepínání mezi user thready bez vzniku problémů s blokováním threadů v jádře a s víceprocesorovými stroji.

Výhoda tohoto modelu je v tom, že uživatel má možnost pouštět velké množství threadů, aniž by tím zatěžoval paměť jádra, a že když thready na sebe budou navzájem čekat, bude jejich probouzení výrazně rychlejší. Pokud je pouštěno velké množství threadů, které budou zablokovány v jádře (například u serveru thready čekající na socketech, posílající nebo čtoucí data od klientů), tak tenhle threadový model oproti ryzím kernel threadům žádné zrychlení nepřinese - thready budou stejně potřebovat kontext jádra (a s ním struct thread), a protože na sebe thready nebudou čekat, tak se výhody threadů v userspace neprojeví.

Pro přepínání procesů umožňuje FreeBSD 5 použít dva algoritmy – původní scheduler z předchozích verzí BSD nebo nový ULE scheduler, který je velmi podobný scheduleru z Linuxu 2.6 (je možno mezi nimi přepnout pomocí options SCHED_4BSD a options SCHED_ULE při kompilaci jádra). ULE scheduler má rovněž fronty pro jednotlivé priority, dvě skupiny front, jednu, ze které se vybírá, a druhou, do které se ukládají procesy, jimž vypršel time slice. Má určování „interaktivních“ procesů a jejich ukládání do aktivní fronty. Podobně jako na Linuxu provádí load-balancing mezi procesory.

9. Adresářový strom

9.1. Stromová struktura

Celou stromovou strukturu adresářů je možné rozdělit na menší části, tzv. svazky. Každá z těchto částí může být umístěna na vlastním disku nebo samostatné diskové oblasti (logické sekci). Tak se lze jednoduše přizpůsobit omezením velikosti disků a zároveň usnadnit zálohování i ostatní úkoly spojené se správou systému. Nejdůležitější z těchto částí jsou adresář -root (kořenový svazek), dále adresáře /usr, /var a /home. Každý systém souborů má jiné určení. Adresářová stromová struktura byla navržena tak, aby fungovala i v síti počítačů s operačním systémem Linux. Uživatelé a programy tak mohou pomocí sítě sdílet některé části systémů souborů, a to buď prostřednictvím zařízení určených pouze pro čtení (např. CD-ROM), nebo pomocí sítě se systémem NFS.

Ještě jedno vysvětlení je tu na místě: v textu se setkáte s pojmy adresář a souborový systém. Souborový systém (File System) je způsob uložení dat na disku (a práce s nimi). Řadu souborových systémů nepochybně znáte (ext2, vfat, iso9660 apod.) Adresář je vlastně již jakousi „podmnožinou“ v souborovém systému. Je to označení pro konkrétní oblast disku, obsahující určitá data.

Pod pojmem Souborový systém si představte disk (nebo jeho logickou část – zkrátka partition). Adresář se již vztahuje ke struktuře uložení dat na tomto disku. Vlastní soubory jsou pak obsahem daného adresáře.

Nejprve je třeba si říci, že **Kořenový adresář -root** (kořenový svazek) je specifický pro každý počítač. Obecně je uložen na lokálním disku. Kořenový svazek obsahuje soubory nutné pro zavedení systému a jeho uvedení do stavu, ve kterém mohou být připojené ostatní souborové systémy. Obsah kořenového souborového systému postačuje pro práci v jednovýživatelském režimu. Na tomto svazku jsou rovněž uloženy nástroje pro opravy poškozeného souborového systému a pro obnovení ztracených souborů ze záloh.

Nyní se podívejme na jednotlivé adresáře:

- Adresář **/usr** obsahuje všechny příkazy, knihovny, manuálové stránky a jiné soubory, jejichž obsah se nemění a které uživatel potřebuje při běžném provozu. Žádný ze souborů svazku /usr by neměl být specifický pro některý počítač. Rovněž by se neměl při normálním provozu měnit. Tyto podmínky zaručují, že soubory uložené v souborovém systému /usr bude možné efektivně sdílet v síti. Sdílení tohoto svazku je výhodné jak z hlediska nákladů, tak z hlediska usnadnění správy systému (např. při instalaci novější verze aplikace se pak mění pouze systém /usr na hostitelském počítači a ne na každé stanici zvlášť). Je-li adresář /usr na lokálním disku, může být připojen pouze pro čtení. To snižuje pravděpodobnost poškození systému souborů při havárii systému.
- Adresář **/var** obsahuje soubory, které se v čase mění. Tedy především sdílené adresáře pro elektronickou poštu, systém -newsio, tiskárny, tzv. log-soubory, formátované manuálové stránky a dočasné soubory. Historicky bývaly všechny soubory, které jsou nyní uloženy v systému /var, v souborovém systému /usr. To znemožňovalo připojit svazek /usr pouze pro čtení.
- Adresář **/home** obsahuje domovské adresáře uživatelů. Vyčlenění uživatelských domovských adresářů do vlastní adresářové stromové struktury nebo samostatného souborového systému ulehčuje zálohování. Ostatní části adresářového stromu totiž buď nevyžadují zálohování vůbec, nebo, vzhledem k tomu, že se nemění tak často, se zálohují jenom zřídka. Velký

adresář /home je dobré rozdělit na několik menších částí hierarchicky nižší úrovně a rozlišit je jménem, např. /home/students a /home/staff.

- Ü Adresář **/tmp** je prostor vyhrazený pro pracovní soubory všeho druhu. Pokud máte možnost připojit „jen“ jeden „přídavný“ pevný disk, rozhodně jej vyhradte pro adresář /tmp . Kapacita tohoto disku zároveň představuje omezení velikosti souborového systému /tmp a to pro všechny uživatele – i root se bude muset tomuto omezení podřídit.

Různé části, na které je hierarchická adresářová struktura rozčleněna, jsou zde označeny jako samostatné svazky (souborové systémy). Není ale žádný zvláštní důvod k tomu, aby ve skutečnosti ležely na samostatných oddělených svazcích. Všechny by mohly být nakonec i v jediném souborovém systému. Takové řešení má význam hlavně pro malé jednouživatelské systémy, kdy je prioritou jednoduchost. Celá stromová adresářová struktura může být rozdělena na souborové systémy i jinak. Způsob jejího rozčlenění závisí na tom, jak velký je disk a jak velký diskový prostor bude vyhrazený pro různé účely. Jedinou věcí, na kterou je potřeba dbát, jsou standardní unixová jména. Ty je potřeba zachovat. I když bude adresář /var a /usr ve stejné diskové oblasti, musí být zachována standardní jména, jako např. /usr/lib/libc.a nebo /var/adm/messages. Totéž platí i v případě, že se například přesune adresář /var do adresáře /usr/var a na původním místě přesunutého adresáře bude symbolický link z adresáře /usr/var.

Struktura souborového systému Unixu sdružuje soubory podle jejich účelu. Jenom tak lze zaručit, že budou například všechny příkazy na jednom místě, data na jiném, dokumentace na dalším a tak dál. Alternativou by bylo sdružovat soubory podle toho, ke kterému programu patří. Pak by mohly být například všechny soubory pro program Emacs v jednom adresáři, všechny soubory pro TEX v jiném a podobně. Problém druhého přístupu je v tom, že je velmi obtížné sdílet soubory – adresář určitého programu často obsahuje jak statické soubory, které lze sdílet, tak soubory, jejichž obsah se mění, a ty sdílet nelze. Rovněž by bylo velmi složité dohledávat v rámci celého systému soubory určitého typu, například manuálové stránky aplikací uložené na mnoha různých místech. Programátory by jistě strašila noční můra, jak v takovémto případě vytvořit programy, které by byly schopné v případě potřeby manuálové stránky všech aplikací nalézt.

9.2. Kořenový svazek -root

Kořenový svazek -root by obecně měl obsahovat co nejméně souborů, protože představuje velmi kritické místo. Faktem je, že u malého souborového systému, který se mění jenom zřídka, je menší pravděpodobnost poškození. Poškození souborového systému -root většinou znamená, že operační systém na tomto svazku nebude možné zavést. Tento problém lze řešit pouze pomocí speciálních opatření, např. zavedením systému z diskety, a „speciální opatření“ obvykle správci příliš nemilují. Není se čemu divit.

Obecně by kořenový adresář systémového svazku neměl obsahovat žádné soubory, snad kromě standardního obrazu systému. Ten se obvykle jmenuje vmlinuz. Všechny ostatní soubory by měly být uloženy v podadresářích kořenového adresáře, obvykle tímto způsobem:

- Ü **/bin** – příkazy potřebné pro zavedení systému a pro práci běžných uživatelů po jeho zavedení.
- Ü **/sbin** – stejné jako u adresáře /bin. Příkazy tímto podadresáři ale nejsou určeny běžným uživatelům, i když je též mohou použít.
- Ü **/etc** – konfigurační soubory specifické pro daný počítač.
- Ü **/root** – domovský adresář superuživatele.
- Ü **/lib** – sdílené knihovny pro programy v kořenovém souborovém systému.

- Ü **/lib/modules** – zaváděcí moduly jádra systému (zvláště ty, které jsou potřeba pro zavedení systému při zotavení po neočekávaných událostech (např. síťové ovladače a ovladače pro souborový systém).
- Ü **/dev** – speciální soubory.
- Ü **/tmp** – dočasné soubory. Programy, které se spouští až po zavedení systému, by správně měly používat místo adresáře /tmp adresář /var/tmp, protože je velmi pravděpodobné, že leží na větším disku.
- Ü **/boot** – soubory, jež používá zavaděč operačního systému (angl. bootstrap loader), např. LILO. Je dobré mít v tomto podadresáři uloženy obrazy jádra (místo toho, aby se ukládaly přímo v kořenovém adresáři). V případě, že jich máte víc, může obsah adresáře /boot značně narůst. V tom případě bude lepší mít jej v samostatném souborovém systému. Tím se také zajistí, že obrazy jádra budou uloženy na prvních 1024 cylindrech disku IDE.
- Ü **/mnt** – přípojný místo pro dočasná připojení dalších systémů souborů správcem systému. Nepředpokládá se, že by tento adresář využívaly pro automatická připojení souborových systémů programy. Adresář /mnt může být rozdělen na podadresáře (např. /mnt/dosa pro disketovou mechaniku používanou souborovým systémem MS.DOS, /mnt/exta pro tutéž mechaniku využívanou v souborovém systému ext2 a podobně).

9.3. Adresář /etc

Adresář /etc obsahuje mnoho souborů. Některé z nich jsou standardní a jsou součástí každé distribuce Linuxu. Pokud jde o ostatní, měli byste nejdříve zjistit, ke kterému programu patří, a pak si přečíst manuálové stránky k tomuto programu. V adresáři /etc je uloženo také hodně síťových konfiguračních souborů, které jsou popsány buď přímo v manuálových stránkách k jádru Linuxu nebo je jejich popis u příslušných aplikací.

- Ü **/etc/rc, /etc/rc.d a /etc/rc?.d** - jedná se o skripty a adresáře skriptů, které se spouští při startu, nebo v případě, že se mění úroveň běhu systému. Podrobnější informace najdete manuálových stránkách procesu init.
- Ü **/etc/passwd** – databáze uživatelů systému s položkami, v nichž je uloženo uživatelské jméno i skutečné jméno uživatele, domovský adresář, šifrované heslo a některé další informace. Formát je popsán v manuálové stránce pro program passwd.
- Ü **/etc/fdprm** – tabulka parametrů disketové jednotky. Popisuje jak vypadají různé formáty disket. Používá ji program setfdprm. Více informací uvádí manuálová stránka programu setfdprm.
- Ü **/etc/fstab** – seznamy souborových systémů připojovaných automaticky při startu příkazem mount .a (skriptem /etc/rc, nebo odpovídajícím souborem, jenž se spouští při startu systému). V systému Linux obsahuje rovněž informace o odkládacích oblastech -swapie, které automaticky používá příkaz swapon .a. Podrobnější informace naleznete například v manuálové stránce k příkazu mount.
- Ü **/etc/group** – soubor podobný souboru /etc/passwd, ale místo uživatelů popisuje pracovní skupiny. Podrobnější informace viz manuálová stránka pro soubor group.
- Ü **/etc/inittab** – konfigurační soubor procesu init.
- Ü **/etc/issue** – soubor obsahuje výstup programu getty, který se zobrazí před výzvou pro přihlášení uživatele. Obvykle obsahuje stručný popis systému nebo uvítací hlášku. Obsah určuje správce systému.
- Ü **/etc/magic** – konfigurační soubor programu file. Obsahuje popisy různých formátů souborů,

podle kterých pak program file tyto typy rozpoznává. Více informací najdete v manuálových stránkách pro magic a file.

- Ü **/etc/motd** – tzv. zpráva pro tento den, automatický výstup na terminál uživatele po úspěšném přihlášení do systému. Obsah volí správce systému. Často se využívá pro předávání informací (např. upozornění na plánovaná zastavení systému apod.) všem uživatelům systému.
- Ü **/etc/mtab** – seznam aktuálně připojených souborových systémů. Jeho obsah po zavedení systému a připojení určených souborových systémů prvotně nastavují inicializační skripty, v běžném provozu pak automaticky příkaz mount. Používá se v případech, kdy je potřeba zjistit, které souborové systémy jsou připojené, např. při zadání příkazu df.
- Ü **/etc/shadow** – soubor tzv. stínových přístupových hesel (shadow password) uživatelů v systémech, které mají nainstalovanou podporu systému stínových hesel. Kódovaná stínová hesla jsou z bezpečnostních důvodů přeneseny ze souboru /etc/passwd do souboru /etc/shadow, který může číst pouze superuživatel. Snižuje se tak pravděpodobnost odhalení některého z přístupových hesel při průniku do systému.
- Ü **/etc/login.defs** – konfigurační soubor příkazu login.
- Ü **/etc/printcap** – podobně jako u souboru /etc/termcap, až na to, že soubor je určený pro tiskárny. Odlišná je i jeho syntaxe.
- Ü **/etc/profile, /etc/csh.login, /etc/csh.cshrc** – soubory spouštěné při přihlášení uživatele nebo při startu systému interprety příkazů Bourne shell nebo C shell. Umožňují správci systému stanovit globální nastavení stejné pro všechny uživatele. Viz manuálové stránky k příslušným interpretům příkazů.
- Ü **/etc/securetty** – soubor identifikuje zabezpečené terminály, tedy terminály, ze kterých se může přihlašovat superuživatel. Typicky je v seznamu uvedena pouze virtuální konzola, takže je nemožné (nebo přinejmenším velmi těžké) získat oprávnění superuživatele přihlášením se po modemu nebo ze sítě.
- Ü **/etc/shells** – soubor, jenž uvádí seznam důvěryhodných interpretů příkazů. Příkaz chsh umožňuje uživatelům změnit shell spuštěný při přihlášení, a to pouze na některý z interpretů uvedený v tomto souboru. Proces ftpd, jenž běží na hostitelském počítači a poskytuje služby FTP pro klientské počítače, rovněž kontroluje, zda je uživatelův příkazový interpret uveden v tomto souboru a nedovolí připojit se klientům, jejichž shell v tomto seznamu uveden není.
- Ü **/etc/termcap** – databáze vlastností terminálu. Popisuje, kterými -escape, sekvencemi se řídí různé typy terminálů. Každý program je napsán tak, že místo přímého výstupu -escape sekvence, jež by fungovala pouze s konkrétním typem terminálu, hledá v tabulce /etc/termcap sekvenci, která odpovídá tomu, co chce program na terminálu zobrazit. Pak může většina programů správně obsluhovat většinu typů terminálů. Více informací uvádí manuálové stránky pro termcap, curs-termcap a terminfo.

9.4. Adresář /dev

Adresář /dev obsahuje speciální soubory pro všechna zařízení. Speciální soubory jsou pojmenované podle určitých konvencí. Tyto konvence jsou samozřejmě podrobně popsány, jejich popis můžete nalézt např. v Linuxové dokumentaci o Seznamu zařízení (je to standardní kapitola každé dokumentace o Linuxu). Speciální soubory se vytvářejí v průběhu instalace operačního systému, v běžném provozu pak skriptem /dev/MAKEDEV. Podobný je skript /dev/MAKEDEV.local. Ten upravuje a používá správce systému, když vytváří čistě lokální speciální soubory a linky. Lokální speciální soubory jsou ty, které nejsou vytvořené standardním postupem pomocí skriptu MAKEDEV, typicky například speciální soubory pro některé nestandardní ovladače zařízení.

9.5. Adresář /usr

Adresář /usr bývá často dost velký, protože jsou v něm instalované všechny programy. Všechny soubory v systému /usr jsou obvykle instalované přímo z distribuce systému Linux. Všechny další lokálně instalované programy se ukládají do adresáře /usr/local. To umožňuje správci systému instalovat vyšší verze Linuxu z nové verze distribuce nebo i úplně jiné distribuce operačního systému bez toho, že by bylo potřeba současně instalovat všechny programy znovu. Některé z podadresářů adresáře /usr jsou popsány níže. Podrobnější popis samozřejmě naleznete ve standardu FSSTND.

- Ü **/usr/X11R6** – všechny soubory systému X Window. Soubory pro X nejsou integrální součástí operačního systému z důvodů zjednodušení vývoje a instalace X. Adresářová struktura /usr/X11R6 je podobná stromu, který je vytvořený pod adresářem /usr samotným.
- Ü **/usr/X386** – podobně jako u předchozího adresáře /usr/X11R6, ale pro systém X11 Release 5.
- Ü **/usr/bin** – zde se nacházejí téměř všechny uživatelské příkazy. Některé další příkazy jsou uloženy v adresáři /bin nebo /usr/local/bin.
- Ü **/usr/sbin** – obsahuje ty příkazy pro správu systému, které nejsou potřeba přímo v souborovém systému -root – zde je například uložená převážná většina serverových programů.
- Ü **/usr/man, /usr/info, /usr/doc** – manuálové stránky, informační dokumenty o projektu GNU, případně různé jiné soubory s dokumentací.
- Ü **/usr/include** – hlavičkové soubory pro programovací C. Z důvodů zachování konzistence by měly být spíš v adresáři /usr/lib, ale z historických důvodů jsou umístěny ve zvláštním adresáři.
- Ü **/usr/lib** – datové soubory pro programy a subsystémy, které se nemění. Jsou zde rovněž uloženy některé globální konfigurační soubory. Jméno lib je odvozeno od anglického slova -library (knihovna). Původně totiž byly adresáři /usr/lib uloženy knihovny podprogramů.
- Ü **/usr/local** – místo pro lokálně instalovaný software a další soubory.

9.6. Adresář /var

Systém /var obsahuje data, která se při běžném provozu systému mění. Soubory jsou specifické pro každý systém, a proto se data mezi jinými počítači v síti nesdílejí.

- Ü **/var/catman** – vyrovnávací paměť pro manuálové stránky, které jsou formátovány na požádání. Zdroje pro tyto manuálové stránky jsou obvykle uloženy v adresáři /usr/man/man*. Manuálové stránky v předem formátované verzi jsou uloženy v adresáři /usr/man/cat*. Manuálové stránky je běžně třeba při prvním prohlížení formátovat. Formátované verze jsou pak uloženy právě v adresáři /var/catman. Další uživatel, který si chce stejné stránky prohlížet, tak nemusí čekat na jejich opakované formátování. Soubory v uvedeném adresáři /var/catman je obvykle potřeba mazat, stejně jako dočasné soubory v adresářích /tmp a /var/tmp.
- Ü **/var/lib** – soubory, které se při normálním provozu systému mění.
- Ü **/var/local** – měnící se data pro programy instalované v adresáři /usr/local, tj. programy instalované správcem systému. Upozorňujeme, že lokálně instalované programy by měly používat i ostatní podadresáře nadřazeného adresáře /var, např. /var/lock.
- Ü **/var/lock** – soubory tzv. zámeků. Většina programů dodržuje určitou konvenci a vytváří v adresáři /var/lock zámky. Tím dávají ostatním programům najevo, že dočasně využívají

některé zařízení nebo soubor. Jiné programy, které by chtěly stejné zařízení či soubor ve stejném okamžiku používat, se o to nebudou pokoušet.

- Ü **/var/log** – adresář obsahuje tzv. log-soubory různých programů. Například program login zaznamenává do souboru /var/log/wtmp všechna přihlášení a odhlášení uživatelů systému, program syslog ukládá do souboru /var/log/messages všechny hlášky jádra systému a systémových programů. Velikost souborů v adresáři /var/log dost často nekontrolovaně roste. Proto se musí v pravidelných intervalech mazat. Ne, řekněme raději "rotovat". Blíže se dozvíte v man logrotate:-)
- Ü **/var/run** – adresář, do něhož se ukládají soubory obsahující informace o systému, jež platí až do jeho dalšího zavedení. Tak například soubor /var/run/utmp obsahuje informace o současně přihlášených uživateli systému.
- Ü **/var/spool** – adresáře pro elektronickou poštu, systém -news, tiskové fronty a další subsystémy, které využívají metodu -spoolij a princip řazení úloh do fronty. Každý z těchto subsystémů má v tomto adresáři svůj vlastní podadresář, např. poštovní schránky uživatelů jsou uloženy v podadresáři /var/spool/mail.
- Ü **/var/tmp** – do adresáře /var/tmp se ukládají velké dočasné soubory a dočasné soubory, které budou existovat déle než ty, které se ukládají do adresáře /tmp. Avšak správce systému by měl dbát na to, aby stejně jako v adresáři /tmp, ani v adresáři /var/tmp nebyly uloženy velmi staré dočasné soubory.

9.7. Adresář /proc

Systém souborů /proc je vlastně imaginárním souborovým systémem. Ve skutečnosti na disku neexistuje. Místo toho jej paměti vytváří jádro systému. Ze systému souborů /proc lze získávat různé aktuální informace o systému, původně o procesech. Některé z významnějších souborů a adresářů jsou popsány níže. Samotný adresář /proc je podrobněji popsán na manuálové stránce proc. Pokud si budete jádro (kernel) překládat sami, nezapomeňte začlenit podporu pro /proc.

- Ü **/proc/1** – adresář s informací o procesu 1. Každý z procesů má v adresáři /proc vlastní podadresář, jehož jméno je stejné, jako identifikační číslo procesu.
- Ü **/proc/cpuinfo** – různé informace o procesoru. Například typ, výrobce model, atd.
- Ü **/proc/device** – seznam ovladačů zařízení konfigurovaných pro aktuálně běžící jádro systému.
- Ü **/proc/dma** – informuje o tom, které kanály DMA jsou právě využívány.
- Ü **/proc/filesystems** – souborové systémy konfigurované v jádru systému.
- Ü **/proc/interrupts** – informuje o tom, která přerušení jsou využívána, i o historii žádostí o využití každého z nich.
- Ü **/proc/ioports** – informuje o tom, který z vstupně-výstupních portů se momentálně využívá.
- Ü **/proc/kcore** – obraz fyzické paměti systému. Má velikost odpovídající velikosti fyzické paměti systému. Ve skutečnosti ale samozřejmě nezabírá takovéto množství paměti, protože jde o soubor generovaný na požádání, tedy pokaždé jenom v okamžiku, kdy k němu různé programy přistupují. Uvědomte si, že soubory souborového systému /proc nezabírají ve skutečnosti (než je zkopírujete na nějaké jiné místo na disku) vůbec žádný diskový prostor.
- Ü **/proc/kmsg** – výstupní hlášení jádra systému. Zde uložená hlášení využívá i program syslog.
- Ü **/proc/ksyms** – tabulka symbolů jádra systému.
- Ü **/proc/loadavg** – statistika zatížení systému – tři celkem nic neříkající indikátory toho, kolik

práce systém momentálně má.

- Ü **/proc/meminfo** – informace o využití paměti, jak fyzické, tak virtuální (swap).
- Ü **/proc/modules** – informuje o tom, které moduly jádra jsou právě zavedeny v paměti.
- Ü **/proc/net** – informace o stavu síťových protokolů.
- Ü **/proc/self** – symbolický link do adresáře procesů toho programu, který zrovna přistupuje k souborovému systému /proc. Když k systému souborů /proc současně přistupují dva různé procesy, budou mít přidělené dva různé linky. Tímto způsobem se mohou programy pohodlně a jednoduše dostat k vlastnímu adresáři.
- Ü **/proc/stat** – různé statistiky týkající se systému. Např. počet chyb stránkování během zavádění systému a podobné.
- Ü **/proc/uptime** – informuje o tom, jak dlouho systém běží.
- Ü **/proc/version** – verze jádra systému.

Většina výše popsaných souborů má textovou formu. Jsou tedy celkem dobře čitelné pomocí standardních nástrojů. Avšak velmi často jsou formátovány takovým způsobem, že informace nich obsažené jsou pro běžného uživatele na pohled dost těžce stravitelné. Proto existuje mnoho příkazů, které (kromě toho, že čtou informace obsažené v souborech systému /proc, upravují jejich formát do srozumitelnější podoby. Tak například program `free` čte data ze souboru /proc/meminfo, převádí velikost v bajtech na kilobajty a přidá něco málo dalších informací o využití paměti.

10. Další vývoj systému

10.1. Budoucnost FreeBSD

10.1.1. Management projektu

Cesta od verze 4 k verzi 5 byla velmi dlouhá a trnitá, a proto bylo rozhodnuto, že se změní způsob vydávání FreeBSD. Dosud bylo FreeBSD uvolňováno dle principu "když jsou hotové featury, tak to stabilizujeme a uvolníme", tento způsob se ukázal jako nevhodný. Uživatelé 4.x čekali velmi dlouho na 5-STABLE, ta stále nebyla k dispozici, a tak bylo mnoho uživatelů (kvůli novým driverům atp.) nuceno užívat testovací verze z řady 5. Bylo zřejmé, že je nutná změna. Ta přišla těsně po vydání 5.3R (která je počátkem stabilní řady 5). Uvolňování dalších verzí bude napříště probíhat dle pevného časového harmonogramu, a to tak, že minor verze budou vydávány každé čtyři měsíce a nové major verze každých 12 - 18 měsíců. Ano, hodně změn se dá sloučit do 5-STABLE, ale ne všechny a obecně to ztěžuje práci. Zajímavé bude, jak team FreeBSD vývojářů (kterých je přesně 214) zvládne větší množství souběžně podporovaných verzí. Dnes jsou tři, bývaly dvě. Nebudeme nuceni do častých upgradů major verzí? Ovšem vzhledem k tomu, že dnes jsou nově vydávány takzvané "Errata Branches", kde se opravují nejen bezpečnostní, ale i další chyby, je možné, že se tohle bude řešit přes "Extended support period", které jsou updatovány nejméně 24 měsíců. Časový plán uvolňování nových FreeBSD pro nejbližší budoucnost vypadá zhruba následovně:

Ü Červen 2005

FreeBSD 5.5R – další z řady 5

Ü Červenec 2005

RELENG_6 branch v CVS repository, která odstartuje stabilizaci vedoucí k uvolnění 6.0R

Ü Srpen 2005

6.0R – první stabilní verze z řady 6, další budou následovat každé čtyři měsíce, 7.0R se očekává 18 měsíců poté

Další novinkou je změna distribuce. Chystá se oficiální podpora šíření přes síť bittorrent (což by mohlo pomoci přetíženým ftp serverům). Revolučnější je ovšem plán vytvoření LiveCD, které bude současně i instalačním médiem, počítá se i se změnou instalátoru. Průkopníkem v tomto směru je projekt FreeSBIE.

10.1.2. Technické cíle

Celým smyslem vytvoření řady 5 byla snaha o lepší podporu SMP strojů. Jako způsob lepší škálovatelnosti byl zvolen systém protekce dat pomocí mnoha mutexů, tedy stejně jako v Linuxu (mnozí s tímto přístupem nesouhlasí a z této rozepry mimojiné vznikl velmi zajímavý projekt DragonFly BSD. Projekt jemného zamykání byl nazván SMPng, v jeho rámci bylo zamknuto mnoho subsystémů, ovšem pracovalo se nejvíce na zamknutí síťování. Zde zazářil Robert Watson, který se svým projektem Netperf dodělal skoro celé zamykání TCP/IP stacku. Projekt Netperf pořád pokračuje, ovšem jeho cíle jsou z velké části splněny. Při inspekci síťového stacku se ovšem přišlo na spoustu nedokonalostí (FreeBSD má nejstarší TCP/IP stack na planetě, zde se TCP/IP zrodilo) a k odstranění těchto vad byl ustanoven projekt Dingo, který si klade za cíl vyčištění a zefektivnění síťového stacku. Práce se zdárně rozbíhají a už kolují patche na efektivnější tcp packet reassembly či nové wifi drivery. Rozhodně je v oblasti síťování (kde se FreeBSD i dnes řadí mezi absolutní špičku) na co se těšit. Jedná se o takové vylepšení jako například prefetching

hlaviček packetů, přepsání ARP kódu a routování, zrychlení zamykání v UMA alokátoru (kde se alokují mbufy pro packety), transakční tcp verze 2 (vlastní vynález FreeBSD - připravuje se RFC) či již zmíněné nové reassemblování tcp packetů. Už teď (s novinkami v řadě 5, jako je netisr či fastforward) je FreeBSD velmi zajímavé a budoucnost vypadá ještě slibněji.

Jednou z oblastí, kde FreeBSD trochu zaostává, je správa dat. A proto byla vytyčena jako jedna z oblastí dalšího vývoje. Konkrétně se jedná o podporu journalování (ne kvůli konzistenci dat - tu ve FreeBSD zajišťuje systém SoftUpdates - ale kvůli zkrácení fsck při bootu), přepsání io scheduleru (existuje implementace anticipatory io scheduleru pro 4.7R, ale nikdy nebyla integrována), podporu clustered FS (podpora SAN řešení) či vyčištění CAM/SCSI kódu. Chystá se také pomalý přechod na gpt partitiony a rozšíření "easy yet powerfull" g* utilit (graid3, gmirror atd.). Poul Henning-Kamp během svého ročního placeného angažmá při vývoji FreeBSD udělal mnoho pozitivních, jež velmi ulehčí další vývoj.

10.1.3. Další technické cíle

Ve verzi 5 byl uveden nový scheduler – SCHED_ULE, jeho autor Jeff Roberson ovšem trochu zanedbal vývoj a betatesting, a tak se stalo, že v 5.3R je ULE v tak špatné kondici, že musel být zakázán. Ovšem Jeff se k vývoji vrátil a ULE v 6-CURRENT funguje velmi dobře.

Vývoj pokračuje na všech frontách – dokumentace je soustavně vylepšována (chystá se její automatické generování doxygenem), man stránky právě podstoupily kosmetické úpravy, co se týče vzhledu, dokonce i msun knihovna podstoupila update (přibýly i386 optimized floated funkce). Nejnovější přírůstek mezi FreeBSD-supported firewally - PF - se dočká updatu na verzi z OpenBSD 3.7, ovšem jen ve verzi 6.

11. Zdroje použitých materiálů

11.1. Texty

11.1.1. Historie a vývoj operačního systému

Kolektiv autorů, Linux – dokumentační projekt.
Computer Press Praha, 2001

[online] [cit. 2005-04-19]

Dostupné z: <<http://jaknalinux.wz.cz/index.php?idp=4>>

Hrabánek J., Linux [online] [cit. 2005-04-19]

Dostupné z: <<http://site.the.cz/index.php?id=23>>

[online] [cit. 2005-04-19]

Dostupné z: <<http://www.olinuxu.cz/sindex.php?akc=clan&clanek=52>>

Rusek O., Linux – základní pojmy [online] [cit. 2005-04-19]

Dostupné z: <<http://www.gybon.cz/~rusek/vyuka/linux.html>>

11.1.2. Distribuce

Distribuce a balíčkování Linuxu [online] [cit. 2005-04-25]

Dostupné z: <<http://home.tiscali.cz/jiriceks/linux.html#uvod>>

Kelner R., Distribuce Linuxu [online] [cit. 2005-04-25]

Dostupné z: <<http://www.linux.cz/distribuce.html>>

Neilo, Distribuce [online] [cit. 2005-04-25]

Dostupné z: <<http://jaknalinux.wz.cz/index.php?idp=6>>

Neilo, Jakou distribuci [online] [cit. 2005-04-25]

Dostupné z: <<http://jaknalinux.wz.cz/index.php?idp=8>>

11.1.3. Instalace systému

Kuře M., Debian GNU Linux 3.0r1 – Instalační příručka
Computer Press Brno, 2003

Kolektiv autorů, Linux Mandrake 9.0 – Instalační příručka
Computer Press Praha, 2003

Kasprzak J., Linux Red Hat – Dokumentační projekt
Computer Press, 1998

Kolektiv autorů CZLUG, Red Hat Linux 7.0 CZ – Instalační příručka
Computer Press, 2001

Jak si nainstalovat Linux? [online] [cit. 2005-04-25]

Dostupné z: <<http://www.linux.cz/install.html>>

11.1.4. Start a konfigurace systému

Kysela M., Přecházíme na Linux
Computer Press, 2003

Adresáře Linuxu [online] [cit. 2005-04-27]
Dostupné z: <<http://site.the.cz/index.php?id=21>>

11.1.5. Uživatelské rozhraní

Janeček V., Tučňákologie aneb i z BFU bude Guru [online] [cit. 2005-04-27]
Dostupné z: <<http://www.root.cz/clanky/tucnakologie-aneb-i-z-bfu-bude-guru/>>

11.1.6. Systém ovládání souborů

Luděk Skočovský, Principy a problémy operačního systému UNIX [online] [cit. 2005-05-02]
Dostupné z: <<http://www.skocovsky.cz/sko93/23.htm>>

11.1.7. Systém správy paměti

Architektúra Linuxu [online] [cit. 2005-05-02]
Dostupné z: <<http://www.strojsnv.sk/linux/uvod/architek.htm>>

11.1.8. Správa procesů

Patočka M., Porovnání systémů Linux a FreeBSD 3. část [online] [cit. 2005-05-02]
Dostupné z: <<http://www.root.cz/clanky/porovnani-linux-freebsd-3/>>

Patočka M., Porovnání systémů Linux a FreeBSD 5. část [online] [cit. 2005-05-02]
Dostupné z: <<http://www.root.cz/clanky/porovnani-linux-freebsd-5/>>

11.1.9. Adresářový strom

Vymazal M., Adresářové struktury [online] [cit. 2005-05-02]
Dostupné z: <http://www.papouch.com/shop/scripts/html/linux/adr_strukt.html>

11.1.10. Další vývoj systému

Divácký R., Budoucnost FreeBSD [online] [cit. 2005-05-02]
Dostupné z: <<http://www.root.cz/clanky/budoucnost-freebsd/>>

11.2. Obrázky

Obr. 1 – Vývoj Unixu
Zdroj: <http://fermat.eup.udl.es/~cesar/images/unix_networking2.jpg>

Obr. 2 – Linus Towards
Zdroj: <<http://site.the.cz/images/linus.jpg>>

Obr. 3 – Tux
Zdroj: <<http://jaknalinux.wz.cz/img/tux.jpg>>

Obr. 4 – Vývoj OS
Zdroj: <<http://home.nedlinux.nl/~m.smeets/unix/page11.gif>>

Obr. 5 - BSD
Zdroj: <<http://cns.utoronto.ca/~pkern/stuff/bsd-gifs/bsd-big.png>>

Obr. 6 – Debian

Zdroj: <<http://jaknalinux.wz.cz/img/debian.jpg>>

Obr. 7 – Mandrake

Zdroj: <<http://jaknalinux.wz.cz/img/mandrake.jpg>>

Obr. 8 – Open

Zdroj: <<http://www.linux.cz/pics/dist/caldera.gif>>

Obr. 9 – Red Hat

Zdroj: <<http://jaknalinux.wz.cz/img/redhat.jpg>>

Obr. 10 – Slackware

Zdroj: <<http://jaknalinux.wz.cz/img/slackware.jpg>>

Obr. 11 – SuSE

Zdroj: <<http://jaknalinux.wz.cz/img/suse.jpg>>

Obr. 12 – Turbo

Zdroj: <<http://www.linux.cz/pics/dist/turbolinux.gif>>

Obr. 13 – Gentoo

Zdroj: <<http://www.linuxguiden.no/images/3/32/Gentoo-logo.png>>

Obr. 14 – Danix

Zdroj: <http://www.danix.cz/portlet_danix.png>

Obr. 15 – Knoppix

Zdroj: <<http://www.knopper.net/pics/knoppix-logo.gif>>

Obr. 16 – Morphix

Zdroj: <http://runlevel5.nm.ru/picture/morphix_logo.jpg>

Obr. 17 – Movix

Zdroj: <<http://movix.sourceforge.net/themes/Kaput/images/logo.gif>>

Obr. 18 – Slax

Zdroj: <<http://slax.linux-live.org/slax.png>>

Obr. 19 – Jedna z možných variant vzhledu startovací nabídky LILO

Zdroj: <http://ftp.upjs.sk/pub/linux/noviny/2001-08/soutez/lilo_boot3.png>

Obr. 20 – Jedna z možných variant vzhledu startovací nabídky GRUB

Zdroj: <<http://www.linuxvirgins.com/images/grub.gif>>

Obr. 21 – Základní obrazovka systému

Zdroj: <<http://i.iinfo.cz/r/photos/bfu/01.png>>

Obr. 22 – Kontextové menu

Zdroj: <<http://i.iinfo.cz/r/photos/bfu/02.png>>

Obr. 23 – Panel zástupců

Zdroj: <http://i.iinfo.cz/r/photos/bfu/08_panel.png>

Obr. 24 – Panel pracovních ploch

Zdroj: <http://i.iinfo.cz/r/photos/bfu/09_panel.png>

Obr. 25 – Panel minimalizovaných oken

Zdroj: <http://i.iinfo.cz/r/photos/bfu/10_panel.png>

Obr. 26 – Panel zamknutí a vypnutí

Zdroj: <http://i.iinfo.cz/r/photos/bfu/11_panel.png>

Obr. 27 – Panel stavových ikon

Zdroj: <http://i.iinfo.cz/r/photos/bfu/12_panel.png>

Obr. 28 – Panel hodin

Zdroj: <http://i.iinfo.cz/r/photos/bfu/13_panel.png>

12. Příloha – prezentace